TRABALHO DE GRADUAÇÃO

INTERSAFE – SEGURANÇA VEICULAR INTERATIVA

Por,

DIOGO CIAN NAZZETTA

Brasília, Setembro de 2013

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

INTERSAFE – SEGURANÇA VEICULAR INTERATIVA

Diogo Cian Nazzetta

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Dr. Ricardo Zelenovsky (Orientador) _____

Prof. Dr. Carlos Humberto Llanos Quintero _____

Prof. MSc. Jones Yudi Mori _____

FICHA CATALOGRÁFICA

NAZZETTA, DIOGO
INTERSAFE – Segurança Veicular Interativa ,

Distrito Federal: 2013.

xi, 39p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2013). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1.Segurança 2.Automotiva
3.Alarme 4.Celular

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

NAZZETTA, D. C., (2013). INTERSAFE – Segurança Veicular Interativa. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 09/2013, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 39p.

CESSÃO DE DIREITOS

AUTOR: Diogo Cian Nazzetta.

TÍTULO DO TRABALHO DE GRADUAÇÃO: INTERSAFE – Segurança Veicular Interativa

GRAU: Engenheiro ANO: 2013

_____

Diogo Cian Nazzetta
EQN 410/411 Bloco A ap 34 – Asa Norte.
70865-405 Brasília – DF – Brasil.

Dedico este trabalho aos meus pais e à minha irmã, que durante toda minha vida me apoiaram em todos os meus projetos pessoais e em minhas loucuras, além de sempre terem me ajudado e me dado a estrutura e a força para seguir em frente, sem desistir.

Dedico em especial a meu pai, Hernando Eduardo Nazzetta, que me ensinou e sempre me incentivou a querer ser uma pessoa cada vez melhor e a não me contentar com o razoável, apenas com a excelência.

Em memória do meu avô, Hernando Francisco Nazzetta, que sempre foi e sempre será uma inspiração em minha vida.

AGRADECIMENTOS

Agradeço à Universidade de Brasília pela acolhida durante todo este período de graduação, bem como meu orientador, Ricardo Zelenovsky, pelo apoio na realização deste trabalho. Agradeço ao meu pai, Hernando Nazzetta, pelos ensinamentos, à minha mãe, Gladys Nazzetta, que com tanto carinho sustentou minha caminhada até este ponto e se preocupou com minha graduação tanto quanto eu. Também devo gratidão a minha querida irmã, Daniella Nazzetta que com grande companheirismo dividiu comigo momentos de angústia e de alegria e que sempre esteve ao meu lado desde o início de minha existência. Por fim, agradeço pelo apoio, pelas ideias e pelas críticas, aos meus amigos próximos, em especial a Vanessa Gomide, cujo apoio e auxilio durante o desenvolvimento e produção deste trabalho foram tão importantes, a Walter Bittar, quem tanto me apoiou nos períodos iniciais da graduação e sem o qual minha permanência em Brasília talvez não tivesse se concluido, e a Ivan Vilela, que foi meu maior companheiro durante as aulas, projetos e as longas noites de estudo.

La semplicità è l'ultima forma della sofisticazione.

Leonardo di ser Piero (da Vinci)

RESUMO

Este texto apresenta o estudo sobre o desenvolvimento, programação e implementação de um sistema de segurança automotiva com funções celulares para interação usuário/veículo. Um sistema baseado em microcontroladores e um módulo celular foi desenvolvido para criar um sistema de segurança veicular que possua uma maior interação com o usuário e que seja mais prático do que os sistemas atualmente disponíveis no mercado, além de eliminar a necessidade de um controle para ativação/desativação do alarme, travamento/destravamento das portas do carro e abertura do porta-malas. Todas esas funcões poderão ser acionadas através do aparelho celular do usuário, independente da marca, modelo ou Sistema Operacional.

Palavras- chave: Segurança, Automotiva, Alarme, Celular.

ABSTRACT

This paper presents the study on the development, programming and implementation of an automotive security system with cellular functions for user/vehicle interaction. A system based on microcontrollers and a cellular shield (or module) will be developed to create a security system with more interaction with the user and more convenient than the currently available systems, in addition to eliminate de need for an alarm control for activating/deactivating the alarm, lock/unlock the doors or opening the car trunk. All these functions will be available by the touch of a button on the user's cellphone, independent of the brand, model or Operating System.

Keywords: Security, Automotive, Alarm, Mobile.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 NOÇÕES INTRODUTÓRIAS

1.1 CONTEXTUALIZAÇÃO

Hoje em dia possuir um veículo acaba sendo sinônimo de preocupação, tanto com os bens que se encontram dentro dele quanto com o próprio veículo, sem mencionar a preocupação com a própria vida, tendo em vista as possibilidades de sequestros ou de latrocínios, infelizmente cada dia mais comuns nos boletins de ocorrência da Polícia Militar. O aumento contínuo da criminalidade nas cidades acaba transformando a satisfação de possuir um veículo em medo ao estacioná-lo.

Diariamente são noticiados diversos casos de furtos de veículos, de objetos no interior dos mesmos e até de rodas, estepes, macacos, dentre outros itens. Uma rápida pesquisa na *internet* revela como o número de ocorrências de crimes ligados a veículos apenas aumenta em todas as regiões do país, sem contar que o delito com maior número de casos relatados é o de furto de objetos no interior dos veículos.

Dentro da Universidade de Brasília a realidade não é diferente, a criminalidade aumenta cada vez mais e, juntamente com o crescente número de alunos que ingressam na universidade, os estacionamentos ficam cada vez mais lotados e as ocorrências ligadas a veículos aumentam quase que proporcionalmente.

A figura 01 mostra o registro das diferentes ocorrências na UnB em 2011:

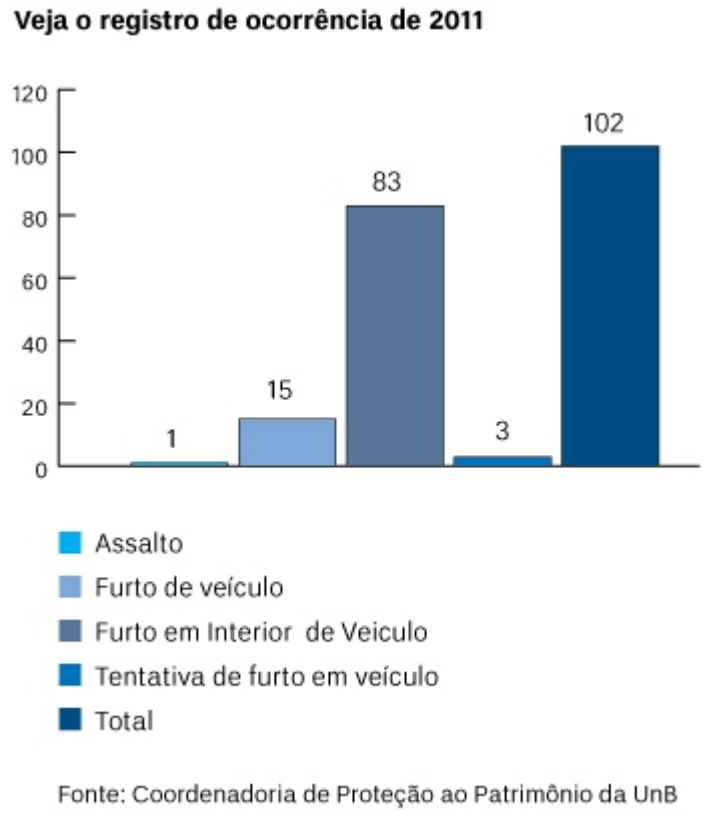


Figura 01 - Registro de ocorrências da UnB

Em 2012, a então gestão do Diretório Central dos Estudantes criou uma ferramenta para relatar e verificar a criminalidade dentro do *campus* Darcy Ribeiro. Seu funcionamento é bastante simples, consiste em um mapa onde as ocorrências são relatadas pelos estudantes (ou por qualquer pessoa vítima de criminalidade dentro do *campus*) e armazendas no próprio site. O mapa evidencia os locais onde foram registradas as ocorrências, criando assim, um "Mapa de Criminalidade" da Universidade (vide “Figura 02”). A intenção do Diretório é que quem visite a página tenha uma noção geográfica do perigo.

Figura 02 - "Mapa de Criminalidade" do *campus* Darcy Ribeiro

A partir da análise do presente tópico, é possível constatar que a ocorrência mais reincidente é a de furto de itens no interior veículo, mais especificamente de Estepes e de Macacos de Elevação.

Diante da exposição de dados supramencionada, mostra-se evidente a necessidade de medidas aptas a coibir ou desencorajar tais práticas criminosas, o que leva à motivação do presente trabalho.

1.2 MOTIVAÇÃO

Foi demonstrado no tópico anterior que o número de furto de veículos é excepcionalmente inferior ao número de furto de objetos no interior do veículo. Destarte, evidencia-se o fato de que é mais cômodo para o malfeitor furtar objetos de menores grandezas.

A motivação principal para desenvolvimento deste projeto veio de uma experiência pessoal vivenciada pelo autor do presente projeto. Na ocasião, o veículo de sua propriedade estava parado em um estacionamento com baixa iluminação e lá permaneceu durante cerca de 3 horas. Ao retornar ao local, encontrou seu carro violado com um dos vidros quebrados e o painel arruinado, devido à tentativa dos meliantes de subtrair o sistema de som. Vários objetos que estavam no porta luvas encontravam-se espalhados pelo interior do automóvel. Apesar dos prejuízos com a janela e com o painel, os vigaristas não foram capazes de levar mais itens do veículo pois, aparentemente, se assustaram com a aproximação de outras pessoas que passavam pelo local e avistaram a cena em que dois homens vasculhavam um carro com alarme sonante, segundo relatos.

Findo o episódio, surgiu a ideia de criar um sistema no qual o usuário pudesse ser notificado no momento em que houvesse o disparo do alarme, podendo assim tomar alguma medida imediata, tendo em vista que muitas vezes o aviso sonoro dos sistemas de alarmes veiculares não é audível pela pessoa a quem mais interessa: o proprietário. Por consequência, o mero alarme sonoro muitas vezes torna-se uma ferramenta meramente ineficaz e inconveniente.

Para evitar a necessidade de portar mais um objeto cotidianamente, houve a ideia de integrar tal função de notificação imediata num aparelho já consagrado e utilizado largamente pela maioria das pessoas: o aparelho celular. Para isso, haveria a necessidade de fazer um sistema baseado em funções celulares que possuísse a capacidade de enviar mensagens SMS e realizar ligações, notificando assim o usuário. Como um módulo celular seria utilizado para tal projeto, estenderam-se as ideias para um sistema de troca de informações bilateral em que se pudesse operar o controle das funções de segurança do veículo através deste módulo.

1.3 OBJETIVOS

Os objetivos deste trabalho consistem em estudar e desenvolver um sistema de segurança veicular interativo com o usuário através de funções celulares e implementá-lo em um veículo cobaia que será utilizado para testes das funções e

para desenvolvimento de novas ideias para o sistema. Este sistema visa uma maior interatividade com o usuário e uma maior praticidade se comparado aos modelos atualmente presentes no mercado.

Posteriormente, objetiva-se a transformação do presente projeto em um produto comercializável, para que qualquer usuário que possua interesse em integrar o referido sistema em seu veículo, possa adquirí-lo. A ideia do sistema é proporcionar uma maior segurança e tranquilidade ao cliente.

1.4 ESTRUTURA DO TRABALHO

O trabalho será apresentado de acordo com a seguinte ordem de capítulos:

A. Noções Introdutórias: Neste capítulo são apresentadas Informações gerais sobre o estudo do trabalho de graduação realizado;

B. Especificidades do Sistema: Capítulo destinado ao detalhamento das ferramentas, peças, módulos e componentes eletrônicos utilizados para o desenvolvimento do presente projeto;

C. Desenvolvimento do Projeto: Neste capítulo são apresentados os trabalhos desenvolvidos até o momento, resultados e expectativas para os trabalhos futuros.

D. Conclusões: Neste capítulo são apresentadas as conclusões do trabalho realizado até agora e as diretrizes das propostas de continuidade dos trabalhos.

E. Bibliografia: Referenciais pesquisados para a elaboração do trabalho desenvolvido

2 ESPECIFICIDADES DO SISTEMA

2.1 ALARME VEICULAR

Um alarme veicular é um dispositivo eletrônico instalado em um veículo, com o objetivo de evitar o roubo do próprio veículo, ou de seu conteúdo, ou de ambos. Alarmes de carro normalmente trabalham emitindo som de alto volume (geralmente uma sirene, buzina, aviso sonoro ou uma combinação destes), quando estiverem reunidas as condições necessárias para o seu disparo. Ademais, o alarme é caracterizado pelo piscar de algumas das luzes do veículo e pela interrupção de circuitos elétricos necessários para a partida do motor.

2.1.1 HISTÓRICO

Uma versão preliminar de alarme veicular a ser utilizado como empecilho para o roubo de veículo foi inventado por um prisioneiro desconhecido de Denver (Colorado, EUA) em 1913.[1] Este sistema era armado manualmente e manifestava-se quando ocorria a tentativa de se acionar o motor.[2] Posteriormente, o projeto de um sistema inspirado nessa versão preliminar, e que utilizava um controle remoto, foi publicado em 1916. Nesta versão, o proprietário do carro deveria transportar um receptor, que emitia um aviso sonoro caso o sistema de ignição do automóvel fosse acionado.

Atualmente, exatos 100 anos após o protótipo supracitado, temos versões de alarmes veiculares que são extremamente eficientes para evitar furto do veículo em si, porém, extremamente ineficientes quando o quesito é proteger objetos no interior do veículo.

2.2 MICROCONTROLADORES

Um Microcontrolador (µC ou MCU) é um circuito integrado programável capaz de executar instruções gravadas em sua memória. É composto por vários blocos funcionais, que realizam tarefas específicas. Um microcontrolador inclui em seu interior as três principais unidades funcionais de um computador:

Unidade central de processamento; memória; e periféricos de entrada/saída. No trabalho em questão será utilizado o microcontrolador AVR ATMEGA328.

2.2.1 AVR ATMEGA328

AVR é um microcontrolador RISC de chip único com uma arquitetura Harvard modificada de 8-bit (µC)[3], desenvolvido pela Atmel em 1996.[4] Foi um dos primeiros da família de microcontroladores a utilizar uma memória *flash* com o intuito de armazenar a programação, diferentemente de seus concorrentes da época, que utilizavam memórias do tipo PROM, EPROM ou EEPROM. A empresa preconiza que o nome AVR não é um acrônimo e não tem nenhum significado em especial. Os criadores nunca se pronunciaram definitivamente sobre o assunto. O ATmega328 é um dos modelos de AVR mais utilizados e mais vendidos no mercado, devido a seu custo reduzido, versatilidade e à sua grande presença nos projetos baseados em Arduino. A “Figura 03” revela o mapa pinagem do AVR Atmega328.

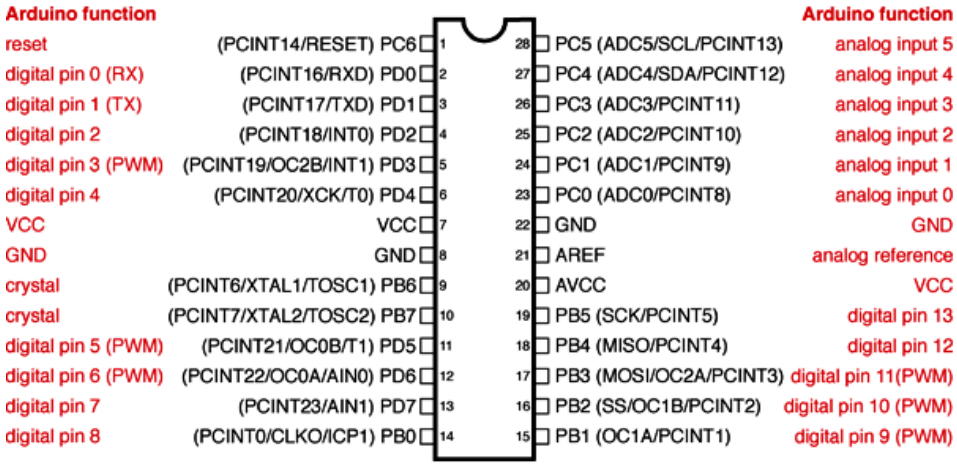


Figura 03 - Pinagem do AVR ATmega328

2.3 ARDUINO

Arduino é uma plataforma de prototipagem eletrônica de placa única *open-source* baseada em *hardware* e *software* flexíveis e de fácil utilização. É destinada a engenheiros, artistas, *designers*, estudantes e a qualquer pessoa interessada em criar objetos, ambientes, ou sistemas interativos.[5] É amplamente utilizada para projetos de automação de diversos sistemas, como residenciais,

industriais, robóticos, dentre outros. Sua programação é feita com sintaxe baseada na linguagem de programação C++ e a programação é mantida na memória do microcontrolador mesmo que a alimentação seja cortada. As placas Arduino possuem um elevado poder de processamento, uma interface bastante amigável, são de fácil programação e possuem um preço bastante acessível se comparadas as outras plataformas de desenvolvimento de projetos para automação.

Existem vários produtos dentro da família Arduino, como Arduino MEGA, Arduino UNO, Arduino DUE (vide “Figura 04”), Arduino Leonardo, Arduino Nano, Arduino Lily Pad, dentre muitas outras. Neste trabalho, será ultilizada a placa Arduino UNO.

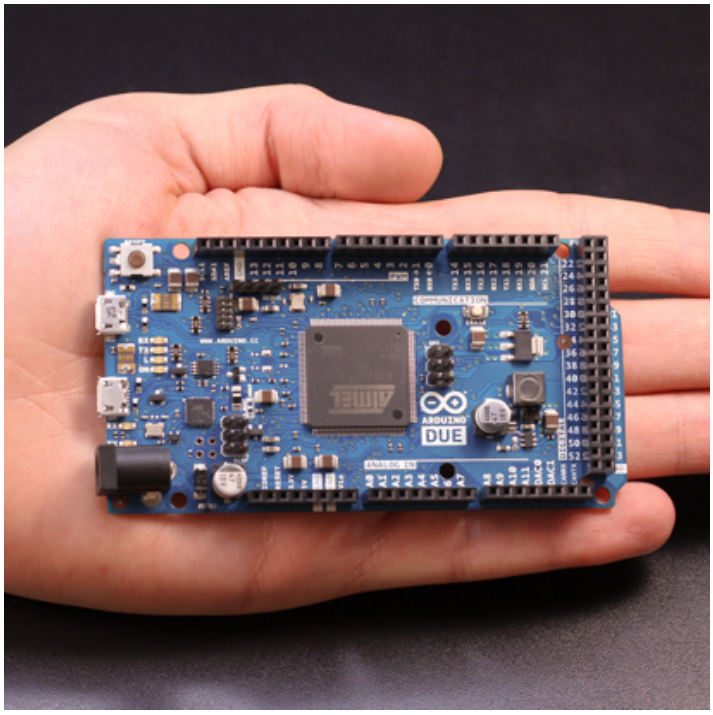


Figura 04 - Exemplo de uma placa Arduino DUE

2.3.1 ARDUINO UNO

Arduino Uno é uma placa baseada no microcontrolador ATmega328. Possui 14 entradas/saídas digitais (das quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um ressonador cerâmico (cristal) de 16 MHz, uma conexão USB, um conector de alimentação, um cabeçalho ICSP (In- Circuit Serial Programmer) e um botão de reset. Ele contém todo o necessário para suportar o

microcontrolador, basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC para DC ou bateria para funcionar.[6]

Atualmente o Arduino encontra-se em sua terceira Revisão (R3), que pode ser visualizado na “Figura 05”.

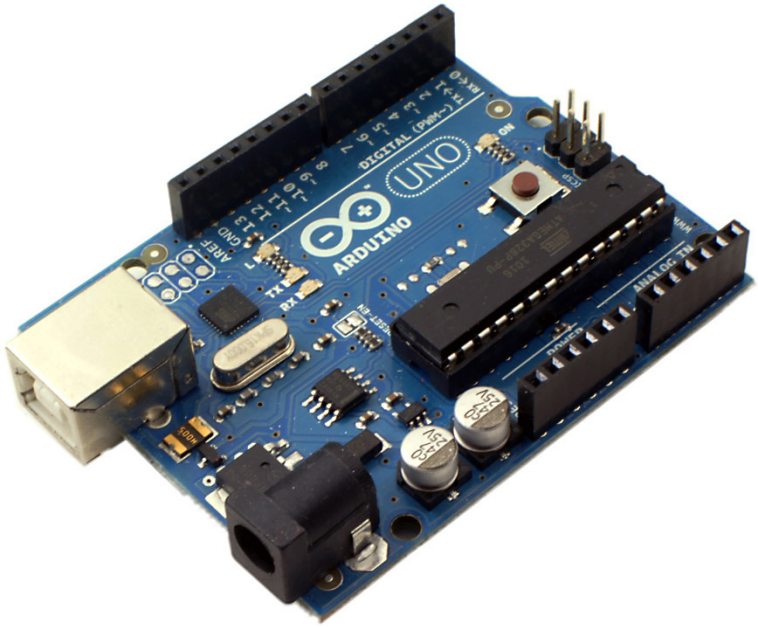


Figura 05 - Arduino UNO R3

O esquemático da placa em questão encontra-se em anexo.

2.4 PROGRAMAÇÃO

A programação do sistema é feita e transferida através do Ambiente de Desenvolvimento Integrado (IDE) do Arduino, disponível para *download* em sua página na *internet* (arduino.cc). A linguagem utilizada para tal é baseada em C++. Os programas (ou *sketchs*, como são chamados) consistem na declaração de bibliotecas, declaração de variáveis globais, Função "*void setup()*", Função "*void loop()*" e demais Funções utilizadas para o projeto.

A função "*void setup()*" é chamada no início de todos os programas de Arduino e executada apenas uma vez, sendo utilizada para:

- Configuração do Arduino;

- Utilização de parâmetros das bibliotecas;

- Inicialização de variáveis;

- Definição do modo de operação dos pinos;

- Definição da velocidade da comunicação serial.

A função "*void loop()*" é uma função executada em ciclo contínuo, cujo objetivo é criar um laço infinito, permitindo que o programa se altere e replique à medida que os parâmetros externos do *hardware* se modifiquem ao longo do tempo, portanto deve-se empregá-la para controlar ativamente o Arduino.

Na “Figura 06” podemos ver o ambiente de desenvolvimento (IDE) com uma parte do código.

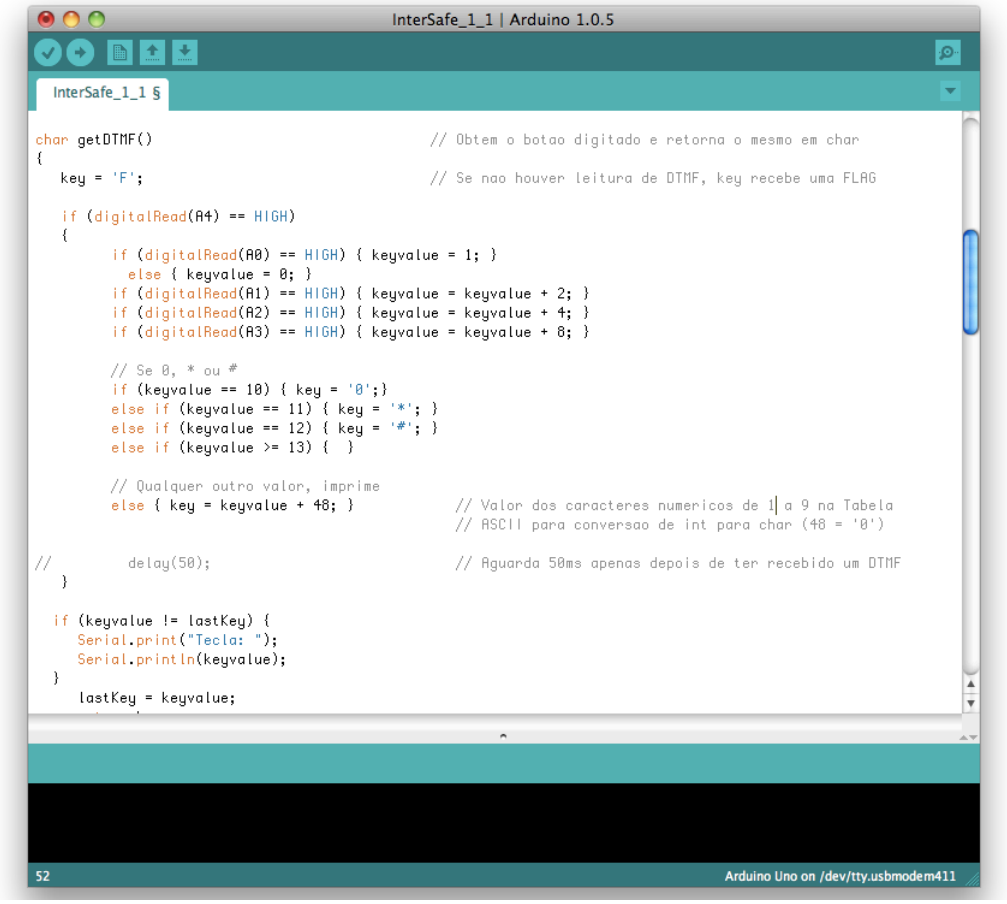


Figura 06 - Ambiente de Desenvolvimento (IDE) com parte do código escrito

A única função deste *sketch* é piscar um *LED*, presente nas placas da família Arduino, normalmente interligada à porta *I/O* de número 13.

2.5 SHIELDS

Shields ("escudos" em inglês) são módulos de fácil conexão que se encaixam no topo das placas Arduino (vide “Figura 07”), possuindo a função de estender as suas capacidades. Recebem este nome pois se assemelham a escudos quando estão conectadas ao Arduino. São bastante amigáveis e fáceis de manusear, tendo em vista que utilizam os próprios pinos conectores elétricos para realizar o encaixe físico, eliminando a necessidade de fios ou conectores entre as placas e possibilitando o encaixe de vários módulos um em cima do outro, diversificando as funções do projeto. Existem diversos modelos diferentes de Shields, possuindo milhares de funções e formatos diferentes, criando-se assim infinitas possibilidades.

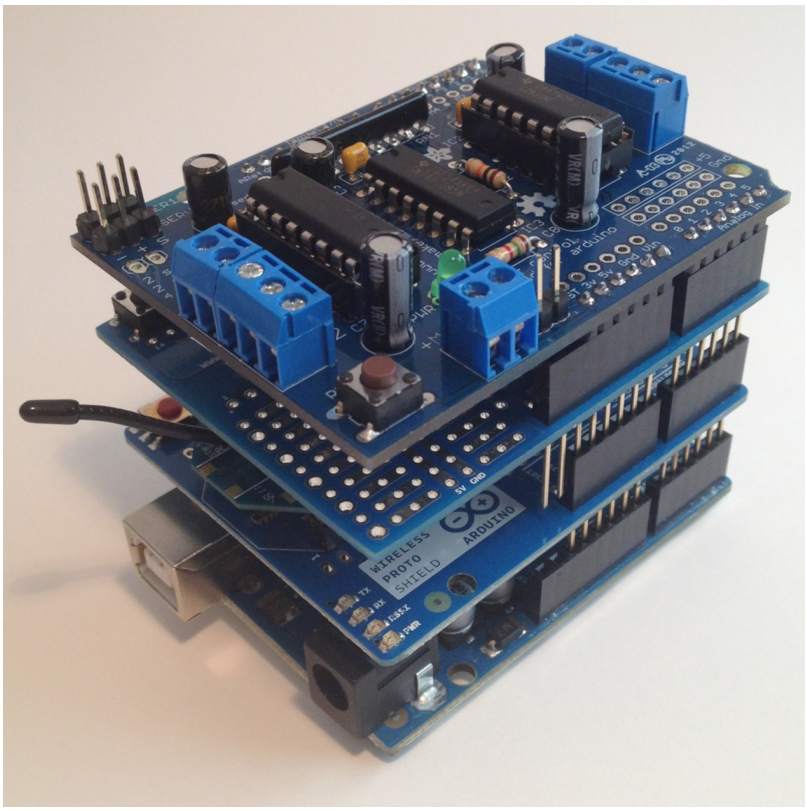


Figura 07- Vários *Shields* empilhados em um Arduino

2.5.1 EXEMPLOS

Ethernet Shield (Figura 08)- Adiciona a possibilidade ao Arduino de conectar-se à *Internet*, enviar *e-mails*, acessar contas de redes sociais, verificar bancos de dados utilizando-se apenas de um cabo de rede comum conectado a um roteador ou mesmo a um computador.

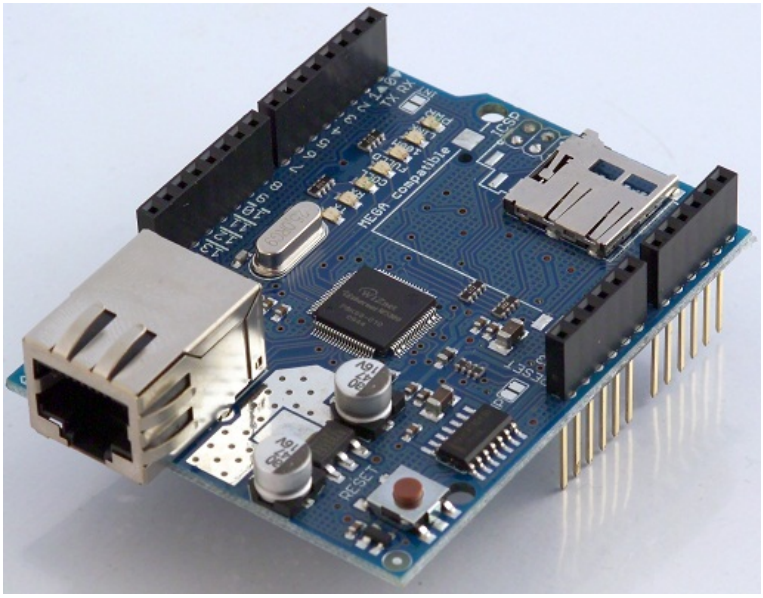


Figura 08 - *Ethernet Shield*

Bluetooth Shield (Figura 09)- Permite a conexão do Arduino a um celular, computador, *videogame* ou diversos outros aparelhos através de tecnologia Bluetooth, permitindo enviar/receber comandos, controlar sistemas e variáveis e realizar troca de informações.

Figura 09 - *Bluetooth Shield*

2.8" TFT Touch Shield (Figura 10)- Adiciona uma tela sensível ao toque de 2.8 polegadas ao Arduino, podendo reproduzir imagens, *menus* sensíveis ao toque e inclusive jogos.

Figura 10 - *2.8" TFT Touch Shield*

2.6 GSM/GPRS SHIELD (Seeed Studio)

O *GSM/GPRS Shield* (Figura 11)funciona de maneira semelhante a um telefone celular comum, podendo executar as mesmas funções básicas além de possibilitar ao Arduino a utilização de tais funções. O *Shield* conta com o módulo *Quad-Band* SIM900, compatível com a grande maioria das empresas de telefonia móvel ao redor do mundo, permitindo a realização e recebimento de ligações de voz, envio e recebimento de mensagens SMS e fax e acesso à *internet* através de tecnologia *GPRS*.

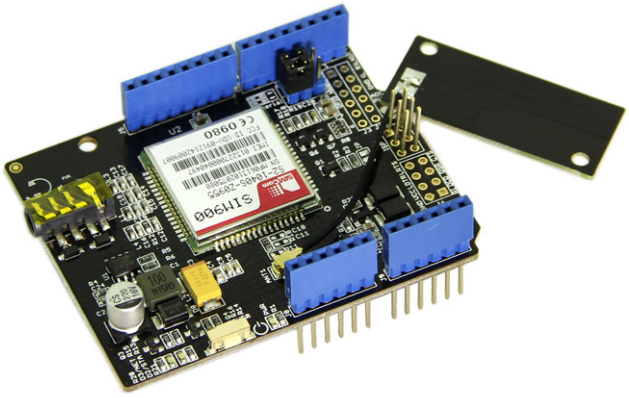


Figura 11 - *GSM/GPRS Shield V2.0 (Seeed Studio)*

Na parte inferior (vide “Figura 12”), o *Shield* possui um *slot* para inserção de um *chip* GSM, o qual será referente a todas as funções executadas. O GSM deve estar ativo e desbloqueado para utilização com o módulo.

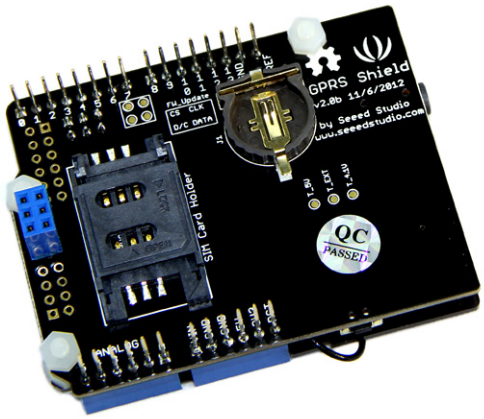


Figura 12 - Parte inferior do *GSM/GPRS Shield*

O módulo é controlado pelo Arduino via comunicação *Serial*, através de comandos na linguagem *Hayes command set* (Conjunto de Comandos Hayes), também conhecida como "*AT COMMANDS*", devido ao fato de todos os comandos serem precedidos pelos caracteres "AT". Esses dois caracteres significam "atenção", pois é uma comunicação baseada em eventos e, por conseguinte, deverá "chamar atenção" para a ocorrência dos referidos eventos. Este padrão de comunicação é amplamente utilizado para controlar módulos a partir de microcontroladores.

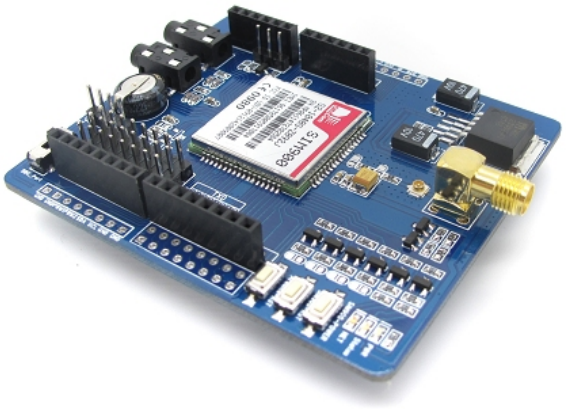


Figura 13 - *IComSat v1.1 (ITead Studio)*

Existem diversos modelos e marcas de *GSM/GPRS Shields* compatíveis com Arduino, inicialmente o módulo utilizado foi o *IComSat v1.1*, da fabricante *ITead Studio* (vide “Figura 13”), pelo valor acessível e facilidade de

aquisição. Porém, ao longo do desenvolvimento do projeto, este foi substituído pelo *GSM/GPRS Shield V2.0*, da fabricante *Seeed Studio*, por sua grande versatilidade e superioridade tecnológica, tendo um menor consumo de energia, maior velocidade e maior confiabilidade do que outros módulos, além de possuir uma antena com melhor capacidade e potência.

Os esquemáticos das duas placas mencionadas acima encontram-se em anexo. Segue no tópico seguinte, a devida explanação sobre a linguagem *Hayes Command Set.*

2.7 **CONJUNTO DE COMANDOS HAYES (AT COMMANDS) - O QUE SÃO?**

O Conjunto de Comandos *Hayes* (*Hayes Command Set*) é uma linguagem de comandos específicos originalmente desenvolvida para o *Smartmodem 300*, produto da empresa *Hayes Microcomputer Products* em 1981. O conjunto de comandos é composto de uma série de sequências de texto curtas, que se combinam para produzir os comandos completos para operações como a discagem, desligamento e mudança dos parâmetros da conexão. O grande sucesso do produto para o qual foi desenvolvida levou a linguagem a ser rapidamente copiada e incorporada por outras empresas do ramo, popularizando-a largamente. A grande maioria dos modems *dial-up* usam o conjunto de comandos *Hayes* em inúmeras variações.[7]

Após 30 anos de sua criação, a linguagem ainda é amplamente utilizada, mas apesar de o padrão ainda permanecer o mesmo, o número de comandos aumentou muito, sendo acrescidos e adaptados à medida que surgem novas tecnologias e/ou novas funções que necessitam de novos comandos específicos.

Uma lista dos comandos mais pertinentes a este trabalho para o controle do módulo *GSM/GPRS* em questão é apresentada a seguir:

2.7.1 COMANDOS RELEVANTES AO TRABALHO

Tabela 01 – Comandos Relevantes ao Trabalho

COMANDO	FUNÇÃO
ATS0=<n>	Determina o número de vezes que o telefone toca antes do Atendimento Automático. (0 = Desativado)
ATS7=<n>	Determina o tempo máximo (em segundos) de espera para que seja completada a chamada
AT+IPR=<n>	Determina a taxa de transmissão de dados do módulo
AT+CMGF	Determina o Formato de SMS
AT+CLVL=<n>	Determina o Volume de saída de áudio
AT+CDRIND	Notifica quando a chamada de voz é encerrada
ATD<n>	Realiza uma chamada de voz para o número <n>
ATH	Encerra a chamada de voz atual (se houver)
AT+CPML	Retorna o número de mensagens armazenadas
AT+CMGR=<n>	Retorna a mensagem armazenada na posição <n>
AT+CMGD=<n>	Elimina a mensagem armazenada na

	posição <n>
AT+CMGDA	Elimina todas as mensagens armazenadas
AT+CMGS	Utilizada para enviar uma Mensagem SMS
AT+VTS	Gera tons de DTMF
AT+VTD=<n>	Determina a duração dos tons DTMF
AT+CPOWD	Efetua o desligamento da placa

2.8 CI's

Um circuito integrado (CI), também conhecido como um *chip* ou *microchip*, é um pequeno circuito eletrônico de material semicondutor de poucos milímetros quadrados de área no qual os circuitos eletrônicos são fabricados e protegidos no interior de um material plástico ou cerâmico. No encapsulamento existem condutores metálicos devidamente posicionados para fazer a conexão entre o chip e uma placa de circuito impresso.

Serão utilizados os seguintes CI's neste trabalho:

- MT8870: Decodificador DTMF

- 4N25: Acoplador óptico

2.9 DTMF

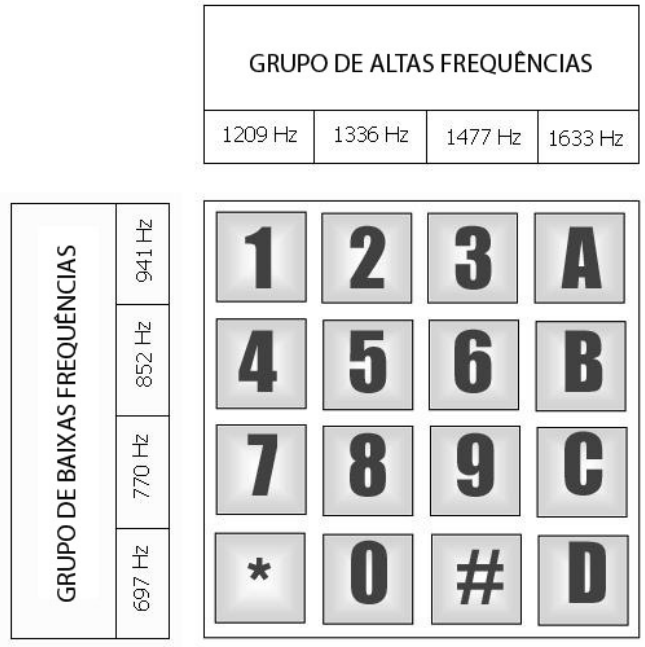
DTMF é a sigla em inglês para "*Dual-Tone Multi-Frequency*", algo como Frequência Múltipla de Duplo Tom. É uma tecnologia de discagem dos telefones mais modernos que utiliza a combinação de duas frequências emitidas ao

mesmo tempo, possibilitando a representação de 16 caracteres diferentes utilizando apenas 8 frequência combinadas duas a duas. Nos primeiros telefones a discagem era feita através de um "disco" que gerava uma seqüência de pulsos na linha telefônica ("discagem decádica" ou "discagem usando sinalização decádica"). Ao se ocupar a linha, o "laço" ("*loop*") era fechado e, ao se efetuar a discagem, ocorriam aberturas periódicas deste "laço", tantas vezes quanto o número discado: para a discagem do 1, uma abertura, para a discagem do 2, duas aberturas, e assim sucessivamente até o 0 (zero) que, na verdade, significava 10 aberturas. Com o advento dos telefones com teclado, das centrais telefônicas mais modernas e com a disseminação dos filtros (primeiro os analógicos, depois os digitais), passou-se a utilizar a sinalização multifreqüencial, uma combinação de tons (os DTMFs vulgarmente conhecidos em inglês por *touch tones*) para discagem.

A sinalização DTMF foi desenvolvida nos laboratórios Bell (*Bell Labs*) visando permitir a discagem DDD, que usa enlaces sem fio como os de micro-ondas e por satélite.

As freqüências destes tons e suas combinações são mostradas na tabela abaixo:

Tabela 02 - *Frequências DTMF*



Na tabela acima são mostradas as freqüências "altas" na linha superior e as baixas na coluna mais à esquerda. No centro o números do teclado. Nos teclados de telefones e celulares são mostrados apenas os números de 0 até 9 e os caracteres " * " e " # ". A freqüência de 1633 *hertz* (e conseqüentemente os algarismos "A", "B", "C" e "D") é utilizada apenas internamente entre equipamentos de teste e medida.

O tom de discagem final, que é enviado à central, é a freqüência obtida pelo batimento da freqüência alta e baixa de uma certa tecla. Por exemplo, para a tecla 5 o tom enviado é a soma de uma senóide na freqüência de 1336 Hz com uma outra senóide de 770 Hz.[8]

Na central, o sinal elétrico é constantemente analisado para detectar a presença simultânea de uma das freqüências baixas e uma das freqüências altas, quando então a tecla do cruzamento destas duas freqüências é identificada pela central.

A escolha destas freqüências se deve principalmente pela baixa probabilidade de se produzir estas combinações de freqüências com a voz humana.

Neste trabalho, utilizaremos os sinais de DTMF para enviar comandos ao dispositivo através do celular, funcionando de forma semelhante a serviços de atendimento ao usuário de bancos ou empresas de telefonia, nas quais usamos o teclado do telefone para digitar senhas ou selecionar opções de um menu.

2.9.1 DECODIFICADOR (MT8870)

O MT8870D/MT8870D-1 é um receptor DTMF completo que integra tanto o filtro divisor de bandas quanto as funções de um decodificador digital. A seção de filtro utiliza técnicas de capacitores comutados para filtragem de grupos alta e baixa frequência. O descodificador utiliza técnicas de contagem digital para detectar e decodificar todos os 16 pares de tom DTMF para um código de 4 bits.[9]

A “Figura 14” mostra um diagrama de blocos do MT8870.

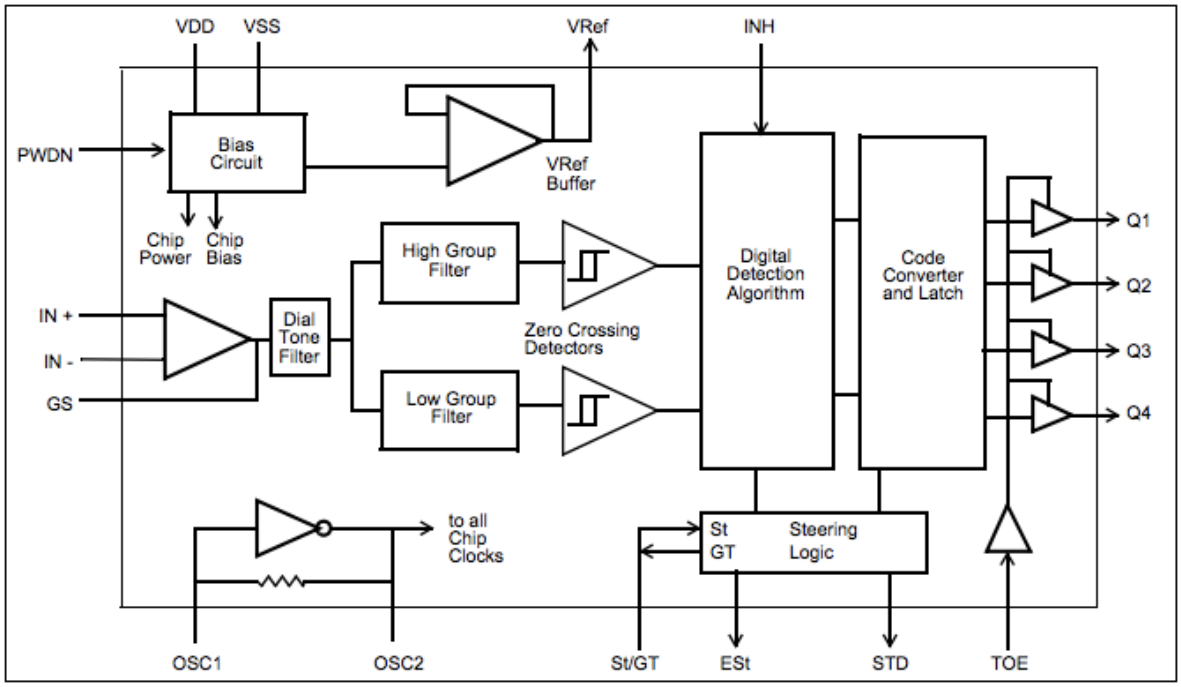


Figura 14 - MT8870: Diagrama de Blocos[9]

A “Figura 15” revela o mapeamento de pinos do MT8870.

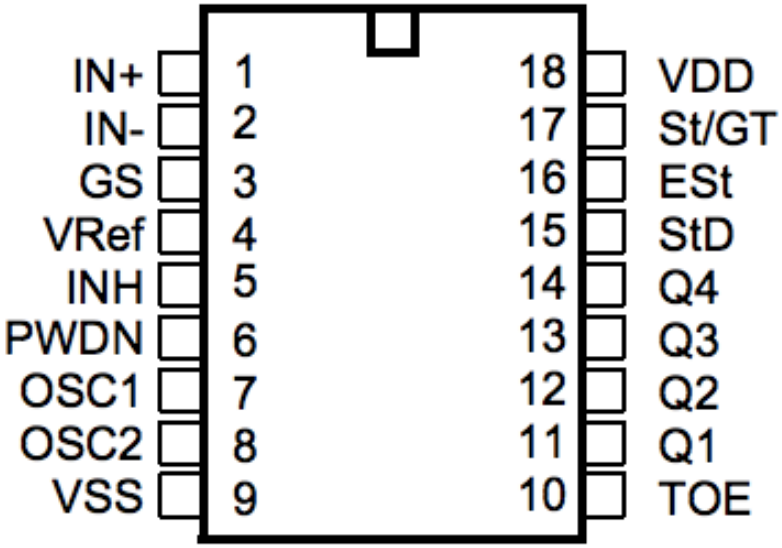


Figura 15 - MT8870: Mapeamento dos Pinos[9]

A tabela abaixo mostra a descrição dos pinos do MT8870.

Tabela 03 - MT8870: Descrição dos Pinos.[9]

Pin Function		
Name	**Function**	**Discription**
IN+	Non-inverting input	Connection to the front-end differential amplifier
IN–	Inverting input	Connection to the front-end differential amplifier
GS	Gain select	Gives access to output of front-end differential amplifier for connection of feedback resistor.
V_{REF}	Reference output Voltage (nominally VDD/2)	May be used to bias the inputs at mid-rail.
INH	Inhibits detection of tones	Represents keys A, B, C, and D
OSC3	Digital buffered oscillator output	
PD	Power down	Logic high powers down the device and inhibits the oscillator.
OSC1	Clock input	3.579545MHz crystal connected between these pins completes internal oscillator
OSC2	Clock output	3.579545MHz crystal connected between these pins completes internal oscillator
V_{SS}	Negative power supply	Normally connected to OV
TOE	Three-state output enable (Input)	Logic high enables the outputs Q1-Q4. Internal pull-up.
Q_1 Q_2 Q_3 Q_4	Three-state ouputs	When enabled by TOE, provides the code corresponding to the last valid tone pair received. (See Figure 2).
StD	Delayed Steering output	Presents a logic high when a received tone pair has been registered and the output latch is updated. Returns to logic low shen the voltage on St/GT falls below V_{TSt}.
ESt	Early steering output	Presents logic high immediately when the digital algorithm detects a recongnizable tone pair (signal condition). Any momentary loss of signal condition will cause ESt to return to a logic low.
St/Gt	Steering input/guard time output (bidirectional)	A voltage greater than V_{TSt} detected at St causes the device to register the dectected tone pair. The GT output acts to reset the external steering time constrant, and its state is a function of ESt and the voltage on St. (See Figure 2).
V_{DD}	Positve power supply	
IC	Internal connection	Must be tied to V_{SS} (for 8870 configuration only).

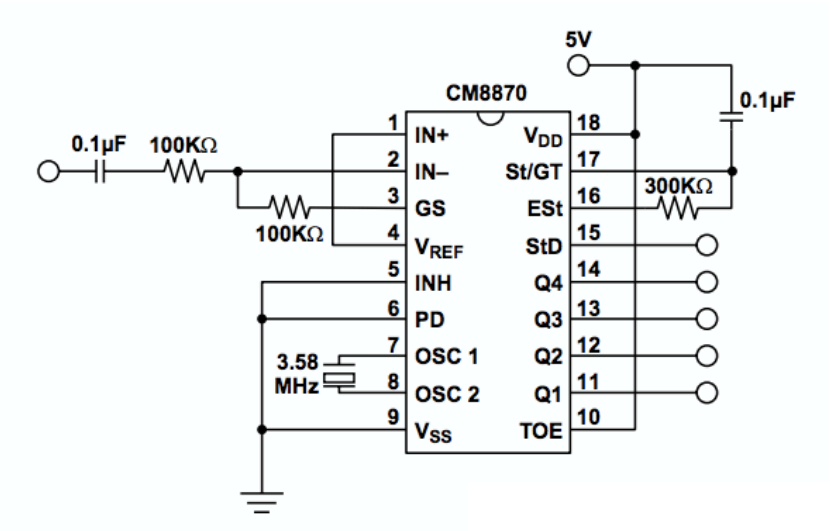


Figura 16 - Circuito típico para o Decodificador de DTMF [9]

A “Figura 16” mostra o Circuito decodificador de DTMF. O circuito recebe o sinal a ser decodificado através das portas 2 e 3 do MT8870 e exibe o

resultado da decodificação no formato binário nas portas Q1, Q2, Q3 e Q4, que mantém os seus respectivos estados até que uma nova decodificação seja realizada. A Saída StD emite um valor lógico alto a cada vez que um sinal é recebido, retornando para o valor lógico baixo quando o sinal é interrompido.

A Tabela a seguir explicita como a decodificação é exibida pelo CI.

Tabela 04 - Tabela de Decodificação Funcional

Functional Diode Table							
F_{LOW}	F_{HIGH}	KEY	TOW	Q_4	Q_3	Q_2	Q_1
697	1209	1	H	0	0	0	1
697	1336	2	H	0	0	1	0
697	1477	3	H	0	0	1	1
770	1209	4	H	0	1	0	0
770	1336	5	H	0	1	0	1
770	1477	6	H	0	1	1	0
852	1209	7	H	0	1	1	1
852	1336	8	H	1	0	0	0
852	1477	9	H	1	0	0	1
941	1336	0	H	1	0	1	0
941	1209	*	H	1	0	1	1
941	1477	#	H	1	1	0	0
697	1633	A	H	1	1	0	1
770	1633	B	H	1	1	1	0
852	1633	C	H	1	1	1	1
941	1633	D	H	0	0	0	0
-	-	ANY	L	Z	Z	Z	Z
L Logic Low, H = Logic, Z = High Impedance							

2.9.2 DTMF DECODER SHIELD

A partir do circuito decodificador de tons DTMF, foi criado o "*DTMF Decoder Shield*", um módulo de decodificação de tons DMTF com base no CI MT8870 para Arduino para ser utilizado em conjunto com o *GSM/GPRS Shield*, na intenção de receber e decodificar os sinais recebidos por este último e repassá-los à plataforma de processamento, para posterior interpretação e tomada de medidas.

3 DESENVOLVIMENTO DO PROJETO

Anteriormente a este projeto, foram desenvolvidos varios outros utilizando diferentes ferramentas de automação, Microcontroladores, CLP's, maquetes e kits de desenvolvimento

3.1 CONCEITO

O Projeto Conceitual consiste em um dispositivo fabricado que responda a funções celulares para controlar os parâmetros de um sistema de segurança veicular. Na visão do usuário, seu telefone celular será utilizado para efetuar uma ligação para o módulo instalado em seu carro (que possuirá um chip GSM e um número de telefone, como qualquer aparelho celular) e, a partir da digitação dos números do teclado, enviará comandos para excução de funções como: Travar ou destravar as portas, abrir o porta-malas, disparar o alarme, dar partida ou desligar o motor, abrir ou fechar os vidros, acionar a buzina ou o farol alto ou o pisca-alerta (com intenção de encontrar o carro em um estacionamento), dentre outras funções, de acordo com a preferência do usuário. Para evitar a violação deste sistema por ação de algum indivíduo de má fé, antes de enviar os comandos pelo teclado do celular, o usuário deverá digitar uma senha de segurança, havendo também a possibilidade de cadastrar alguns números de celular para que o módulo aceite comando APENAS destes números cadastrados. Se desejado, os comandos também podem ser enviados por mensagem SMS.

O módulo será configurado para atender chamadas automaticamente (de qualquer número ou apenas dos números cadastrados, à escolha do cliente) e sua saída de áudio estará conectada à entrada de sinal do circuito decodificador de DTMF. Ao digitar números no teclado de seu celular, o usuário estará enviando tons para o módulo, que por sua vez operará a decodificação e passará então os valores digitados (em binário) para a central de processamento (no caso, o Arduino) que por sua vez fará a interpretação destes valores, verificação da senha e, mediante recebimento dos comandos, enviará sinais para relés ou para o controle RF do alarme do carro, que executarão as funções finais desejadas.

Todo o sistema é alimentado com Tensão DC de 5V. Como a fonte de energia disponível será a bateria do veículo, que possui tensão DC 12V (podendo variar entre 11V e 14V), foi utilizado um circuito de redução com proteção para evitar picos no sistema e o risco de queima dos componentes. Este circuito consiste basicamente de uma fonte chaveada com seus respectivos componentes (resistores e capacitores) programados para fornecer uma redução para 5V e pode ser visto na Figura 17.

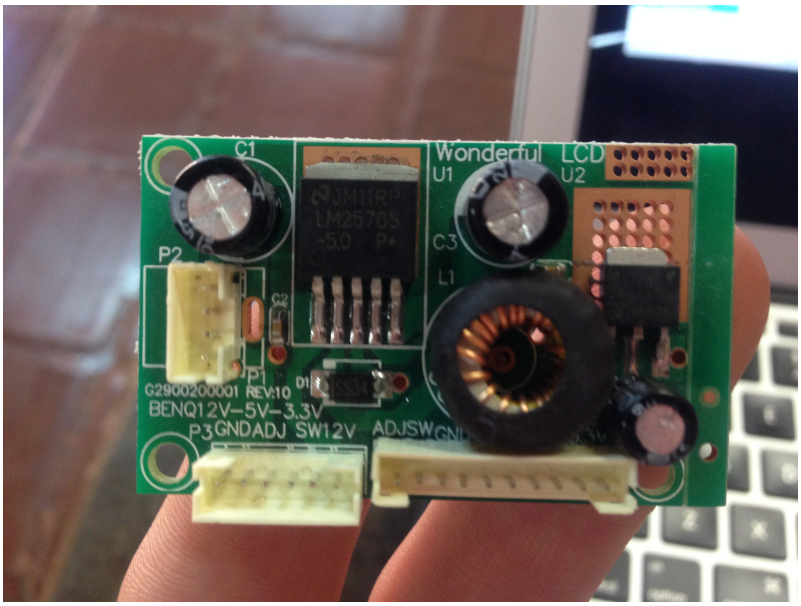


Figura 17 - Circuito de redução de tensão

3.2 MONTAGEM E TESTES INICIAIS

Para os testes iniciais com a placa e primeiros passos, uma montagem simples foi feita, consistindo apenas em um Arduino Uno R3 e o módulo GSM/GPRS *Shield* V2.0 da *Seeed Studio* e uma programação extremamente simples, a partir da qual comandos AT podem ser enviados a partir do computador em tempo real através do *Monitor Serial* da IDE do Arduino, que é utilizado para enviar e receber informações da placa.

A troca de informações entre o Arduino e o *Shield* GSM se dá por comunicação Serial, porém, para não ocupar as portas de número 0 e número 1 do Arduino (RX e TX, respectivamente) que são destinada a este tipo de comunicação,

uma função chamada *SoftwareSerial* é utilizada. A intenção de não ocupar estas portas é permitir a livre comunicação entre o Arduino e um computador, para utilização do *Monitor Serial* para processo de *Debug* ou mesmo para aplicações às quais isso seja interessante.

Os primeiros testes envolveram apenas envios de comandos AT para a placa e observação das respostas recebidas. Ligações foram efetuadas e recebidas manualmente, sem nenhum processo automático.

A partir daí, os primeiros programas testes com funções automáticas foram desenvolvidos baseados em programas exemplo, facilmente encontrados na *internet*.

3.3 PRIMEIROS PROGRAMAS

No primeiro programa desenvolvido, o sistema efetuava, de maneira alternada, ligações para dois números salvos na memória, mediante o pressionamento de um botão, aguardava alguns segundos e desligava a chamada. O Processo se repetia até o desligamento do sistema. A partir deste programa inicial já foi possível fazer estudos sobre atrasos de ligação e tempo necessário para completar ou desligar chamadas.

Posteriormente, um programa que enviava mensagens SMS para vários números salvos na memória foi desenvolvido com o mesmo intuito de estudar parâmetros do sistema implementado. Tais parâmetros foram estudados e levados em conta nos próximos programas escritos. Após alguns outros programas testes, a primeira versão da Programação Final começou a tomar forma.

3.4 CARACTERÍSTICAS DA *GPRS SHIELD*

Como mencionado anteriormente, o Módulo GSM utilizado neste trabalho é o *GPRS Shield V2.0* da *Seeed Studio*, ele possui um *plug* P2 de saída de áudio que é aproveitado para conexão com o módulo DTMF através de um cabo P2 de áudio comum, simplificando a tarefa de transportar o sinal recebido pela rede de telefonia até o módulo decodificador.

Os pinos utilizados pelo *Shield* em questão são apenas os pinos 7 e 8 do Arduino (Rx e Tx no *Shield*, respectivamente) para comunicação *Serial* e o pino 9, que é utilizado para ligar e desligar o módulo através de *Software*. Existe também a possibilidade de utilizar os Pinos 0 e 1 do Arduino para comunicação *Serial*, porém, por preferência do autor, a troca de dados será mantida nos pinos 7 e 8. Portanto, todas os outros pinos estão livres para utilização e manipulação de entradas e saídas I/O.

3.4 TESTES COM DTMF

Para os primeiros testes utilizando DTMF, um circuito baseado no CI MT8870 foi montado e testado. No Datasheet podemos encontrar o circuito a ser montado para realizar as decodificações, como foi mostrado na Figura 16. O primeiro circuito montado contava com LED's nas saídas Q1, Q2, Q3, Q4 e StD, assim como na Figura 18 e era conectado direto à saída de som de um celular.

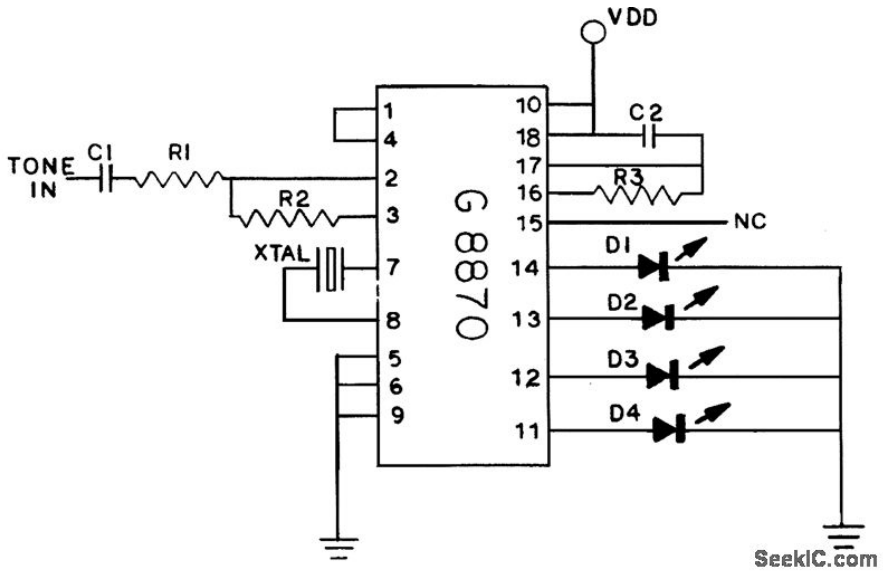


Figura 18 - Decodificador *DTMF* com *LED's*

Após testes e estudos com os LED's, os mesmos foram removidos, os terminais conectados a portas I/O do Arduino e um programa de testes foi desenvolvido para responder a determinados números pressionados no teclado do celular.

Por fim, um pequeno *Shield* foi desenvolvido com o intuito de diminuir a quantidade de fios entre os componentes e aumentar a praticidade, permitindo a montagem de boa parte do sistema apenas encaixando os módulos uns sobre os outros. A figura 19 evidencia o plug P2 do protótipo do *Shield* Decodificador de *DTMF*.

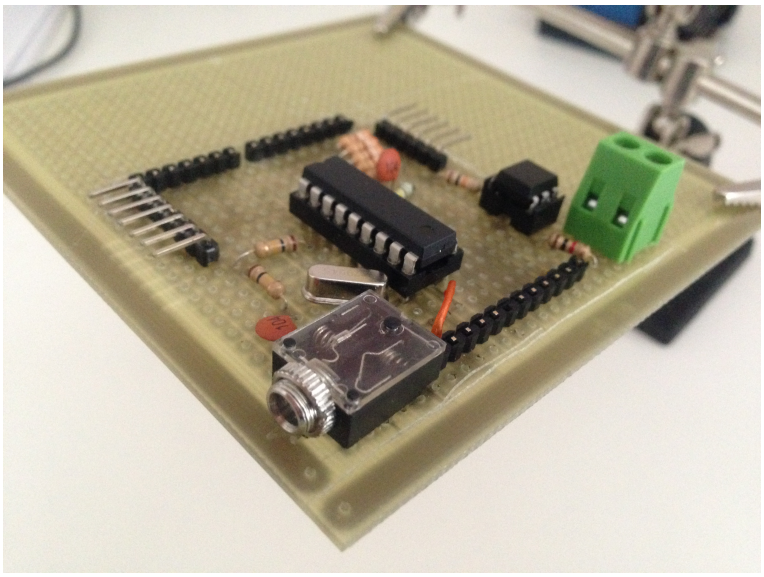


Figura 19 - *Plug P2* do Protótipo do *Shield* Decodificador *DTMF*

Neste *Shield*, um plug de áudio tipo P2 fêmea (3,5mm) foi adicionado para tornar a conexão com o *Shield GSM* mais prática e simples e as saídas Q1, Q2, Q3, Q4 e StD do CI HT8870 são ligadas às portas de entrada Analógica A0, A1, A2, A3 e A4 do Arduino, respectivamente. A “Figura 20” mostra a parte inferior do protótipo do *Shield* Decodificador *DTMF*.

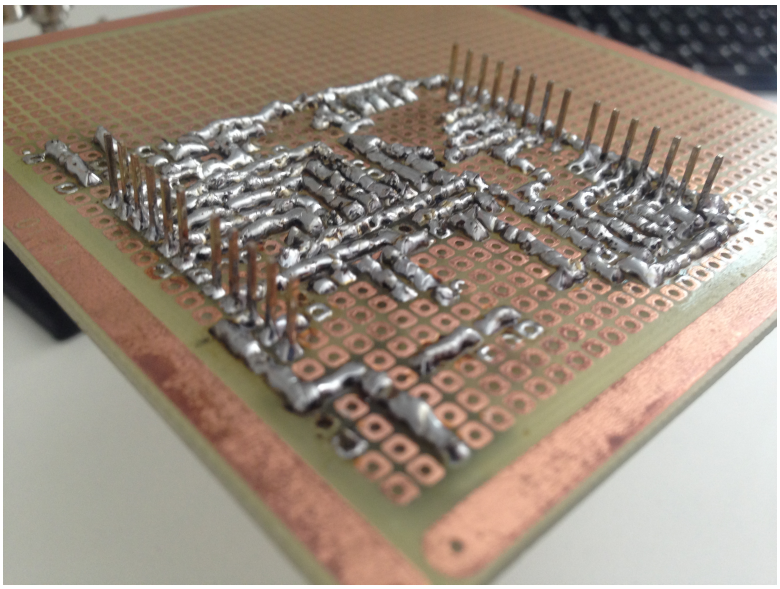


Figura 20 – Protótipo do *Shield* Decodificador *DTMF* - Parte inferior

Após vários testes com o protótipo do *Shield*, uma versão definitiva foi projetada e fabricada, dando uma maior robustez e confiabilidade ao projeto. A Figura 21 mostra o projeto da placa definitiva projetava utilizando o *software* livre Fritzing e a Figura 22 mostra a placa já fabricada.

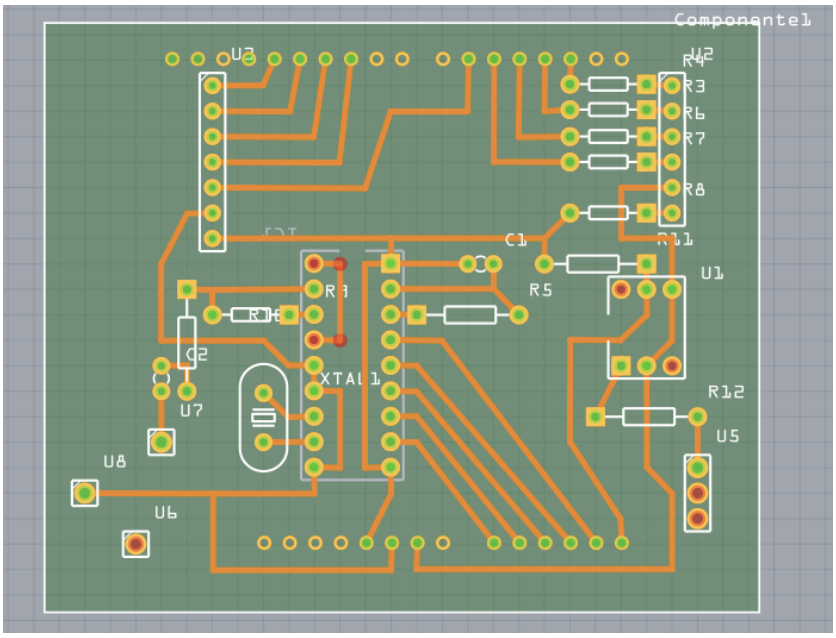


Figura 21 – Projeto da placa do *Shield* Decodificador *DTMF* no *Fritzing*

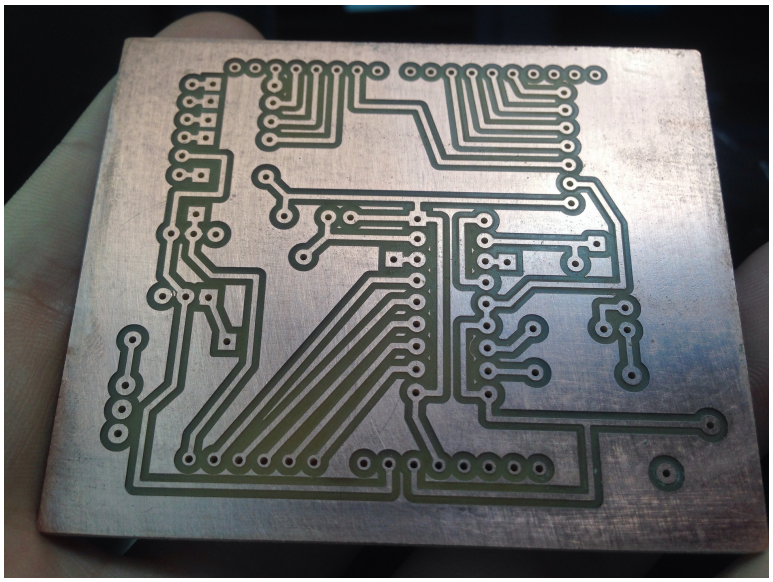


Figura 22 – Placa do *Shield* Decodificador *DTMF* Fabricada

A Figura 23 e a Figura 24 mostram, respectivamente, a parte superior e inferior do Shield já pronto e soldado.

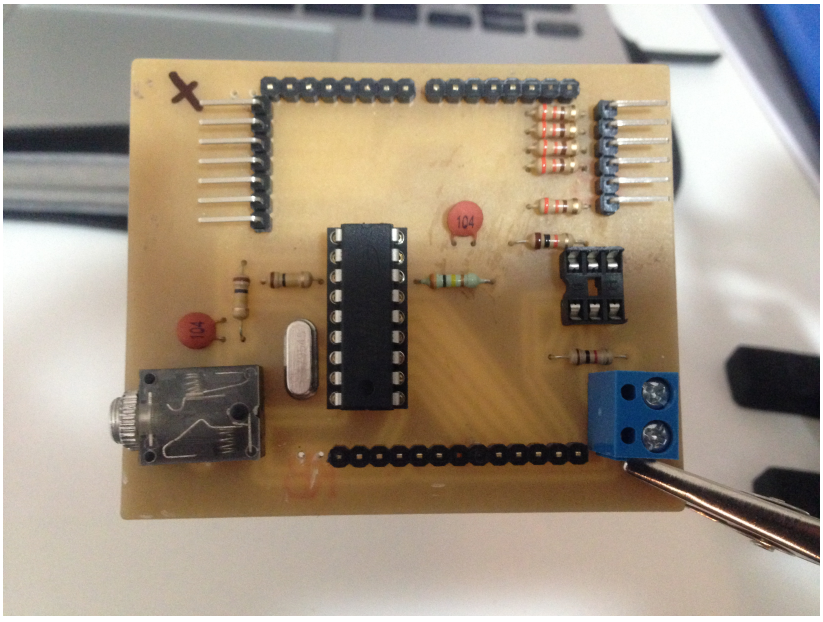


Figura 23 – Versão final do *Shield* Decodificador *DTMF* – Parte superior

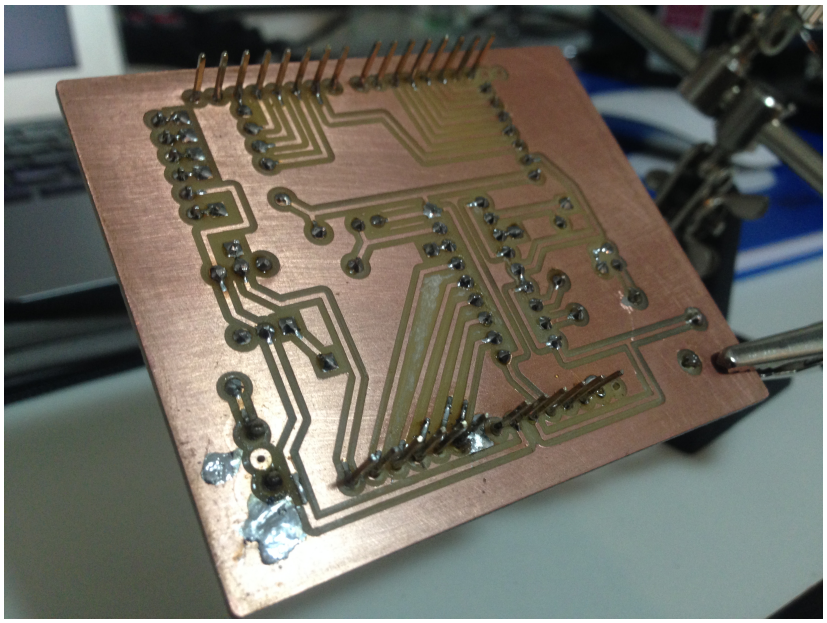


Figura 24 – Versão final do *Shield* Decodificador *DTMF* – Parte inferior

Além disso este *Shield* também foi utilizado para conectar as saídas do microcontrolador aos relés e ao controle de telecomando do alarme do carro. A “Figura 25” mostra o Shield Decodificador DTMF montado.

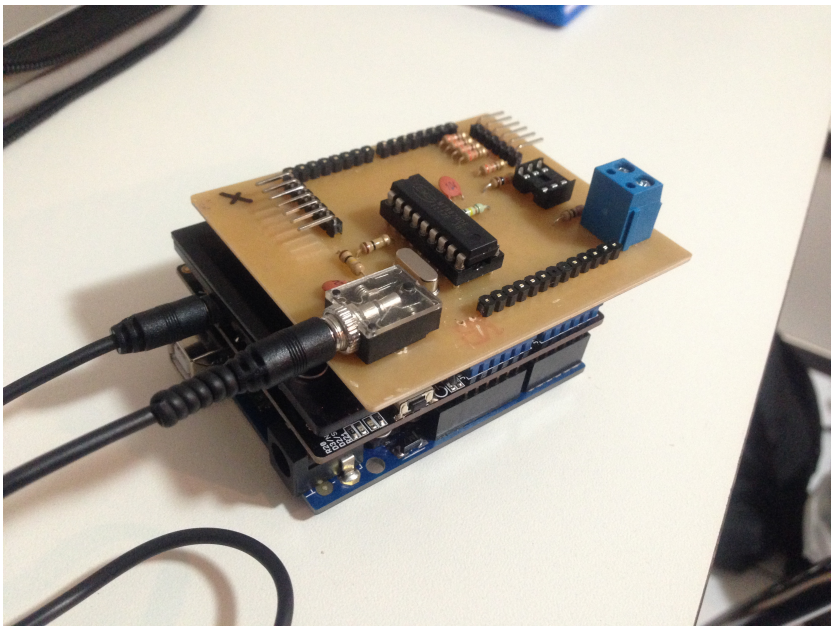


Figura 25 - *Shield* Decodificador *DTMF* montado

3.5 VEÍCULO COBAIA

O veículo utilizado como cobaia para receber o protótipo é um Ford Fusion ano 2012, um carro que possui um moderno sistema eletrônico e diversas proteções contra fraudes e tentativas de roubo/furto, já possuindo um sistema de alarme integrado.

Apesar de não ser exatamente o modelo ideal para receber o sistema de segurança em desenvolvimento deste trabalho (pela quantidade de elementos eletrônicos de proteção já existentes no carro), o grande desafio será acoplar o módulo em questão ao automóvel sem afetar o sistema já existente no carro, apenas complementando-o. Tal fator também irá demonstrar como o projeto em desenvolvimento poderá ser aplicado a, praticamente, qualquer carro, sem limitação de modelo, marca ou nível tecnológico.

Por ser um veículo que ainda está dentro do prazo de garantia, o projeto deverá ser aplicado evitando ao máximo o corte de cabos, pois viola os termos de garantia do fabricante. A implementação se limitará apenas a descascar cabos e se aproveitar dos atributos já existentes no carro para realizar as funções desejadas.

3.6 SISTEMA ELÉTRICO

Como mencionado no tópico anterior, o veículo em questão a ser utilizado possui diversas proteções e não poderá ter seus cabos livremente cortados, portanto, para não alterar o esquema elétrico do carro e evitar a perda da garantia, um dos controles do alarme será utilizado e integrado ao sistema para controlar as funções de alarme, evitando a alteração de esquemas elétricos do carro e isolando um pouco mais os dispositivos do projeto das variações hostis de tensão e corrente que ocorrem com certa frequência no sistema elétrico do veículo. Quanto mais integrados eletricamente estiverem os dois sistemas, maior deverá ser a preocupação com proteções e interferências, portanto é bom que haja esta separação como medida de segurança.

3.7 CHAVE E TELECOMANDO (CONTROLE) DO ALARME

Para o veículo Ford Fusion, o modelo do telecomando do alarme fornecido pelo fabricante está integrado à chave de ignição. Para evitar a destruição de uma chave original para âmbito do projeto, uma nova chave foi encomendada. O preço cobrado por uma peça destas original é exorbitante e chega a ser absurdo em alguns casos. Infelizmente, concessionárias e Chaveiros especializados acabam sendo incoerentes na cobrança de um serviço que pode ser conseguido por até um décimo do valor cobrado por eles.

Uma chave original foi encomendada pela *internet* por cerca de R$ 90,00 (Sedex incluso) e o processo de codificação, tanto do telecomando do alarme, como do *transponder* da chave são extremamente fáceis se o usuário possui a chave original, sem absolutamente nenhum gasto adicional. O serviço do Chaveiro ainda será necessário para fazer a cópia da lâmina da chave, porém não é cobrado mais do que R$ 10,00 para isso.

Tendo isso em vista, a utilização de um telecomando de alarme original para carros que já possuam sistema de segurança integrado torna-se bastante viável, evitando aborrecimentos com a garantia do veículo.

3.8 DESENVOLVIMENTO DO CÓDIGO

Para o desenvolvimento do código final, as primeiras coisas a serem determinadas são as declarações da função "*void setup()*". Todas as declarações de definições do *Shield GSM* devem estar contidas. Nesta função o módulo já tem sua taxa de transmissão de dados determinada, que, para este projeto, foi determinada em 2400 bit/s e o modo de SMS configurado para texto, além de já ser programada para realizar o atendimento automático de qualquer número, para desligar automaticamente após 33 segundos caso uma chamada não seja completada e para manter o volume máximo de saída de áudio pelo *plug* P2. A taxa de transmissão do *Shield* GSM pode variar de 1200 bit/s a 115.200 bit/s, para este projeto, uma taxa de transmissão mais baixa foi escolhida para uma melhor manipulação das informações recebidas pela comunicação *Serial* pelo microcontrolador.

Para o conceito de segurança do código, foi levado em conta a necessidade de uma senha para que unicamente o proprietário tenha acesso às funções vitais do sistema, sendo assim, como o programa roda em *loop*, ele deve sempre retornar ao pedido da senha, pois é o estágio mais importante do código. Uma vez digitada corretamente, um sinal de confirmação sonoro é enviado pelo módulo e o usuário terá 10 segundos para executar alguma função disponível. Caso nenhuma tecla seja pressionada neste período, o programa retorna ao ponto da exigência de senha. Porém, caso o usuário, após a digitação correta da senha, pressione qualquer tecla dentro do período de 10 segundos, a função é executada e o contador zerado novamente, de modo que ele tenha mais 10 segundos para executar outra função.

Para este projeto, as funções esperadas são:

- Travamento das portas

- Destravamento das portas

- Abertura do porta-malas

- Disparo do Alarme Sonoro

- Acionamento da Partida do Motor

- Desligamento do Motor

Porém, outras funções também podem ser adicionadas, como por exemplo:

- Bloqueio e Desbloqueio da Ignição do Motor

- Abertura ou Fechamento dos Vidros

- Abertura ou Fechamento do Teto Solar (Se disponível)

- Acionamento da Buzina

- Acionamento dos Faróis ou Luz Alta

- Acionamento do Pisca-Alerta

- Abertura ou Fechamento da Capota (Se disponível)

Além destas funções que podem ser executadas pelo usuário, este projeto também conta com o modo Notificação de Disparo de Alarme, no qual o módulo efetuará ligações e enviará SMS's ao telefone celular do proprietário em caso de disparo do Alarme, trespassando assim a situação problemática na qual o usuário estaciona seu carro a uma distância em que ele não é capaz de ouvir o Alarme Sonoro. O referido código encontra-se no anexo deste trabalho.

Na Figura 26, podemos ver o fluxograma do produto final.

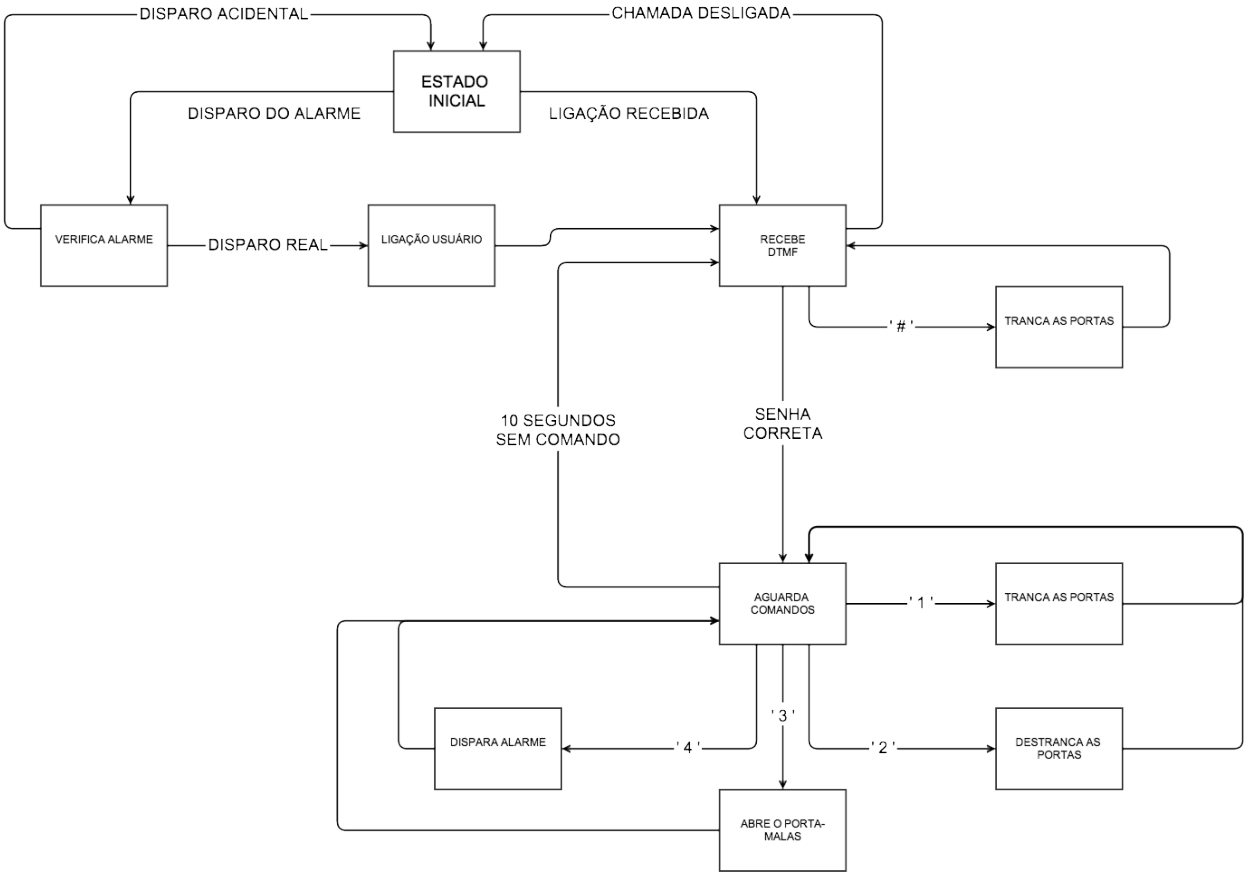


Figura 26 - Fluxograma do produto final

3.9 CUSTOS

A ideia deste projeto e tornar-se um produto comercializável, tendo isso em mente, devem ser levados em conta os custos necessários para a fabricação do mesmo. O custo do protótipo inicial foi relativamente alto, mas caso se torne um produto, estes valores podem ser significantemente reduzidos. Diminuindo ao máximo os gastos, podemos chegar aos seguintes valores para cada componente necessário na fabricação do INTERSAFE:

- Microcontrolador	R$ 15,00
- Módulo GSM/GPRS	R$ 100,00
- Componentes Eletrônicos	R$ 30,00
- Antenas	R$ 10,00

Portanto, temos os custos totais na faixa de R$ 150,00 , o que é um valor bastante competitivo se levado em conta os preços de sistemas de alarme veiculares que não oferecem as mesmas funções deste sistema.

4 CONCLUSÃO

A ideia para o presente trabalho surgiu de um evento recorrente na vida de várias pessoas, especialmente no Distrito Federal, onde o índice de criminalidade ligado a veículos cresce cada vez mais. Ao longo do tempo, tal ideia amadureceu e foi tomando forma com este projeto, assim buscou-se desenvolver um protótipo que visava, principalmente, a informação do usuário no momento em que seu veículo fosse violado, uma das maiores falhas nos sistemas de alarme atuais. Ao utilizar um módulo celular para fazer a notificação do proprietário do veículo, decidiu-se aproveitar suas funções para efetuar o controle total das outras finalidades do sistema, eliminando assim a necessidade de um outro controle exclusivo para isso.

Ao longo do desenvolvimento do projeto, diversas dificuldades foram encontradas, dentre elas o desenvolvimento de um Shield decodificador de DMTF para o Arduino, a programação de todo o projeto, que é um processo extremamente iterativo e envolve muitas variáveis e a implementação do sistema em um carro cobaia, pois envolve a desmontagem do painel e de outras partes do carro e devem ser levados em consideração diversos fatores como picos de tensão e gestão de energia da bateria. Apesar disso, cada uma das dificuldades encontradas foi de grande ajuda na evolução do processo de experiência com sistemas automotivos/sistemas embarcados.

Todos os testes realizados, seja com o Microcontrolador, com o Shield GSM, com o Decodificador de DTMF, com o veículo cobaia, ou com o sistema completo, tiveram, ao final, os resultados esperados, apesar dos contratempos. Algumas proposições tiveram que ser alteradas, devido a limitações do veículo, dos próprios módulos utilizados e inclusive limitações financeiras. Apesar disso, o conceito do projeto permaneceu o mesmo e o objetivo inicial foi alcançado.

Muito foi aprendido com este projeto no âmbito de programação, eletrônica e engenharia em geral. Tal aprendizado visa a materialização de fato deste protótipo inicial na forma de um produto destinado ao público. Trabalhos futuros envolvem a maturação do conceito, desenvolvimento de novas funções e

possibilidades, divulgação do projeto para o público-alvo e muita pesquisa, visando o emplacamento do produto no mercado e possivelmente diminuindo a quantidade de ocorrências de furtos no interior do veículo ou dos próprios veículos nas grandes cidades.

BIBLIOGRAFIA

1 HEARST MAGAZINES. Prisoner Devises Stolen Automobile Alarm. Popular Mechanics, Chicago, abr. 1913.

2 BONNIER CORPORATION. New Automobile Alarm Calls for Help. Popular Science, Chicago, maio 1916.

3 SILVA, Juarez Bento da. Monitoramento, aquisição e controle de sinais elétricos, via Web, utilizando microcontroladores. Disponível em: <http://www.tede.ufsc.br/teses/PGCC0387.pdf>. Acesso em: 01 set. 2013.

4 FARIA, Thiago Henrique Daud de. Introdução aos Microcontroladores. Disponível em: <http://www.lps.usp.br/lps/arquivos/conteudo/grad/dwnld/mcu.pdf>. Acesso em: 01 set. 2013.

5 ARDUINO TEAM. Arduino. Disponível em: <http://www.arduino.cc/>. Acesso em: 02 set. 2013.

6 ARDUINO TEAM. Arduino. Disponível em: <http://www.arduino.cc/>. Acesso em: 02 set. 2013.

7 KRAFTWERK. Hayes Smartmodem. Infoworld, Palo Alto, n. , p.9, jul. 1981.

8 DODD, Annabel Z. The essential guide to telecommunications. Ptr: Prentice Hall, 2002.

9 ZARLINK SEMICONDUCTOR. MT8870D/MT8870D-1 Datasheet. Disponível em: <http://www.futurlec.com/Datasheet/Zarlink/MT8870DS.pdf>. Acesso em: 3 set. 2013.

ANEXO I

```
// ************************************************************** //
// ************************************************************** //
// ***                                                        *** //
// ***                  SOFTWARE INTERSAFE 1.1.1              *** //
// ***                                                        *** //
// ************************************************************** //
// ************************************************************** //


#include <SoftwareSerial.h>
SoftwareSerial mySerial(7, 8);


int onModulePin = 9;                      // Pin to switch on the module
(without press on button)


int keyvalue = 0;                         // Valor referente ao codigo
binario obtido (1-12)
int lastKey;                              // Comparar mudanca de botao
char key;                                 // Caracter obtido no DTMF (0-
9, *, #)
char keyS;
char lastkeyS;
char* senha = "1234";                     // Senha
int position = 0;                         // Posicao de leitura do
codigo digitado referente a senha


unsigned long startTime = 0;              // Tempo inicial para contagem
dos 10s do tempo de comando


// ------------
int Lock   = 4;                           // Portas referentes aos
telecomandos
int unLock = 2;
int Trunk  = 3;
int PANIC  = 5;
// ------------


char phone_number[]  = "0416185009550";    // Celular de Brasilia
char phone_number2[] = "06299653737";     // Celular de Goiania


// ================================================================


void switchModule()                             // Turn on the Module
{
   digitalWrite(13, HIGH);               // FLAG
```

```
   mySerial.println("AT+CPOWD=1");      // Envia Sinal para DESLIGAR a
placa,
   delay(4000);                         // caso ela esteja ligada.

   digitalWrite(13, LOW);               // FLAG
   digitalWrite(onModulePin,HIGH);      //
   delay(2000);                         // Liga a placa atraves do pino
9,
   digitalWrite(onModulePin,LOW);       // OnModulePin
   Serial.println("Placa Ligada");      // Envia Sinal para DESLIGAR a
placa,
}


// ================================================================


char getDTMF()                                  // Obtem o botao
digitado e retorna o mesmo em char
{
  key = 'F';                                   // Se nao houver leitura
de DTMF, key recebe uma FLAG

  if (digitalRead(A4) == HIGH)
  {
        if (digitalRead(A0) == HIGH) { keyvalue = 1; }
          else { keyvalue = 0; }
        if (digitalRead(A1) == HIGH) { keyvalue = keyvalue + 2; }
        if (digitalRead(A2) == HIGH) { keyvalue = keyvalue + 4; }
        if (digitalRead(A3) == HIGH) { keyvalue = keyvalue + 8; }


        // 'Special' char tones
        if (keyvalue == 10) { key = '0';}
        else if (keyvalue == 11) { key = '*'; }
        else if (keyvalue == 12) { key = '#'; }
        else if (keyvalue >= 13) {  }


        // Anything else, print
        else { key = keyvalue + 48; }            // Valor dos
caracteres numericos de 0 a 9 na Tabela
                                                 // ASCII para
conversao de int para char (48 = '0')


//         delay(50);                               // Aguarda 50ms
apenas depois de ter recebido um DTMF
  }


 if (keyvalue != lastKey) {
    Serial.print("Tecla: ");
    Serial.println(keyvalue);
 }
    lastKey = keyvalue;
    return key;
```

```
}

// ====================================================================


void verificaAlarme()
{
 boolean State;
 boolean lastState = HIGH;
 int horns = 0;
 startTime = millis();
 while(millis() - startTime <= 3300 && millis() - startTime >= 0)
 {
   State = digitalRead(A5);
   if(State != lastState)
   {
     if (State == LOW)
       horns++;
     lastState = State;
   }
 if(horns >=4)
   call_User();
 }
}


// ====================================================================


void call_User()
{
 mySerial.println("ATD85009550;");
}


// ====================================================================


void setup()
{
   Serial.begin(2400);                  // Serial Monitor UART baud
rate
   mySerial.begin(2400);                // Shield UART baud rate


   pinMode(13, OUTPUT);
   pinMode(onModulePin, OUTPUT);        // Definicao dos pinos


   pinMode(Lock, OUTPUT);              // Binary 1
   pinMode(unLock, OUTPUT);            // Binary 2
   pinMode(Trunk, OUTPUT);             // Binary 3
   pinMode(PANIC, OUTPUT);             // Binary 4
//    pinMode(2, INPUT);                  // Steering (StD)
```

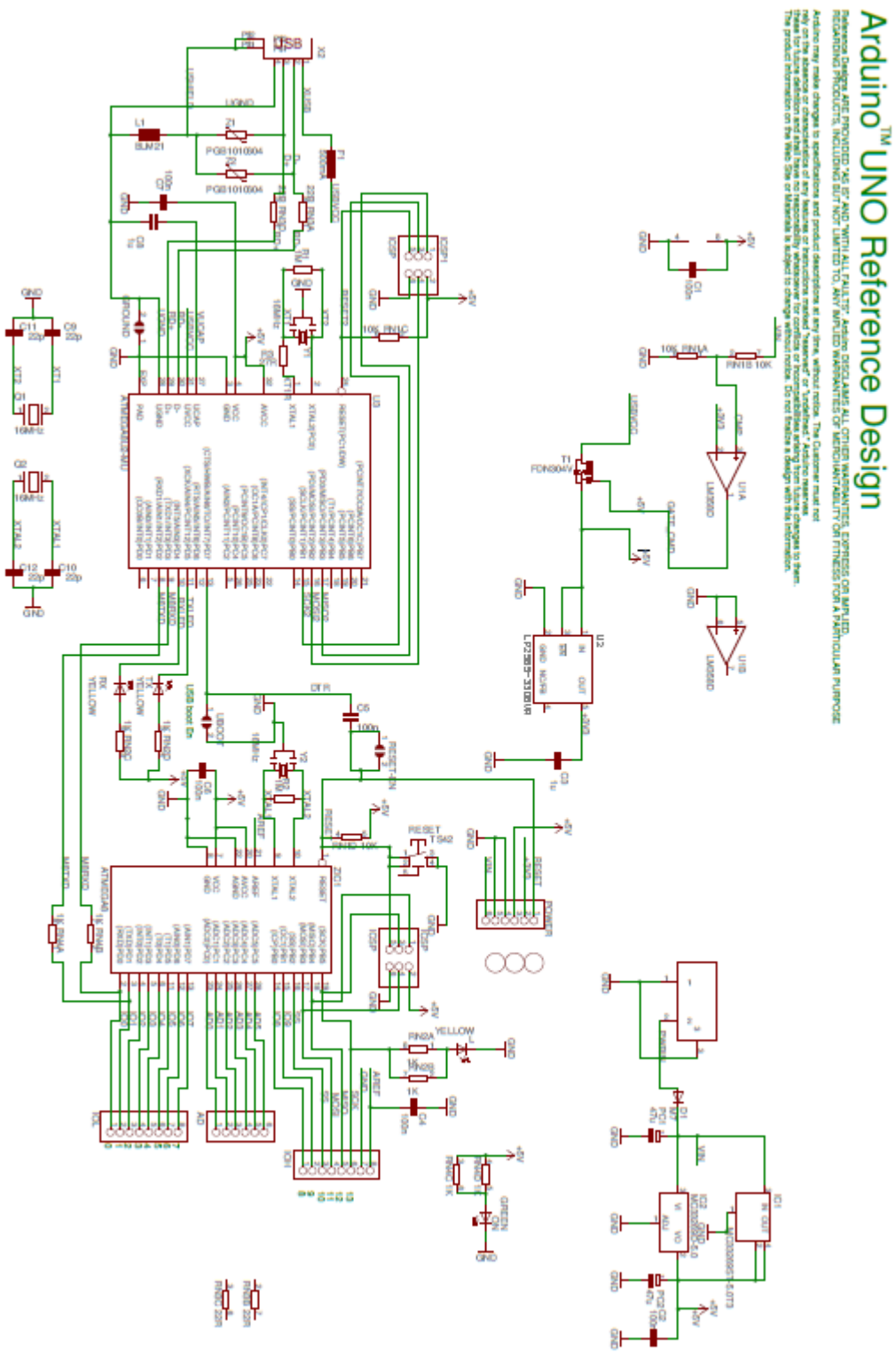
```
   switchModule();                      // Chama a funcao de ligar a
placa

   delay(5000);
                                        // Definicoes da placa:
   mySerial.println("ATS0=1");          // Atende chamadas apos tocar
uma vez
   delay(1000);                         //
   mySerial.println("ATS7=30");         // Desliga uma chamada apos 30
segundos
   delay(1000);                         //
   mySerial.println("AT+CLVL=100");     // Volume 100%
   delay(1000);                         //
   mySerial.println("AT+CMGF=1");       // Define o modo de SMS como
TEXT
   delay(1000);                         //


   mySerial.println("AT+CLDTMF=5,\"0,0\"");  // Geracao local de DTMF (
\" <- Imprime aspas na Serial)
   delay(2000);

   Serial.println("Pronto");
}



// ================================================================ //
//                              LOOP                                //
// ================================================================ //


void loop()
{
 if(digitalRead(A5)==LOW)
     call_User();


 keyS = getDTMF();                  // Chama a funcao getDTMF e obtem
em char o botao digitado
 if (keyS != lastkeyS) {
   if (keyS == '#') {      // Se digitar * ou #, Imprime "Locked"
     position = 0;
     Lockit();
   }

   if (keyS == '*') {      // Se digitar * ou #, Imprime "Locked"
     position = 0;
   }

   if (keyS == senha[position]) {         // Se o char obtido coincidir
com a primeira posicao da senha,
     position++;                          // o contador posicao aumenta
e assim por diante
   }
```

```
   else if (keyS != 'F')                    // Se keyS = 'F', nao vai
coincidir com senha[position], mas
     position = 0;                          // significa que nenhum novo
numero foi digitado, porem se
                                            // keyS != 'F' !=
senha[position], ERROU! comeca de novo!!


   if (position == 4) {                     // Se o contador position
chegar a 4, os 4 digitos da senha
     lastkeyS = keyS;                        // Evita que a repeticao
da leitura do mesmo botao
     teleCommand();                         // foram acertados
     position = 0;
   }
 lastkeyS = keyS;                           // Evita que a repeticao da
leitura do mesmo botao
 }
//    delay(50);                               // Aguarda 50ms antes de
verificar se chegou outro DTMF
}


// ====================================================================


void teleCommand()
{
 mySerial.println("AT+VTS=\"#\"");      // Geracao de DTMF na ligacao (
\" <- Imprime aspas na Serial)
 delay(50);
 Serial.println("SENHA CORRETA !!");
 Serial.println("Favor escolher um comando");
 startTime = millis();
 while(millis() - startTime <= 10000 && millis() - startTime >= 0)
 {
   keyS = getDTMF();
   if (keyS != lastkeyS) {
     switch (keyS) {
       case '1':      //
         Lockit();
         break;
       case '2':      //
         unLockit();
         break;
       case '3':      //
         openTrunk();
         break;
       case '4':      //
         PANICit();
         break;
       case '5':      //
         PANICit();
         break;
       case '6':      //
         Lockit();
         Lockit();
         break;
```

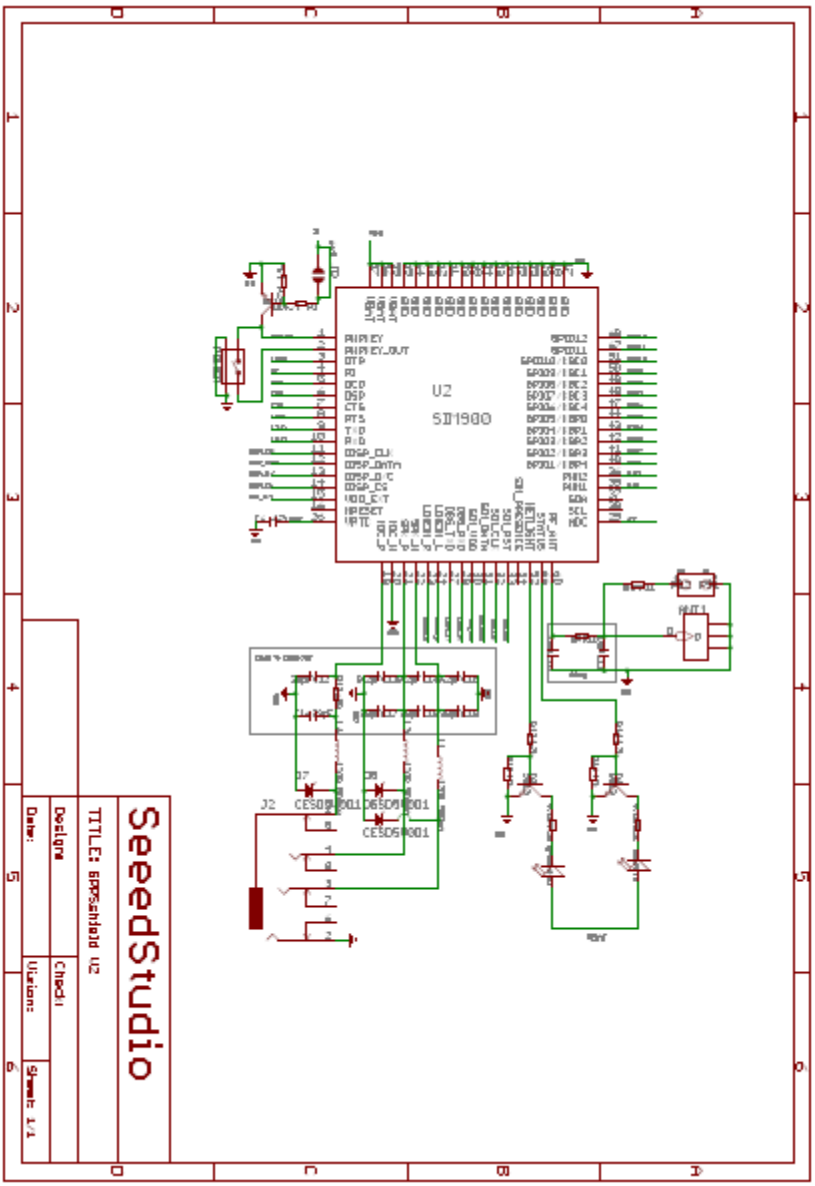
```
       case '7':      //
         unLockit();
         unLockit();
         break;
       case '8':      //
         Lockit();
         break;
       case '9':      //
         Lockit();
         break;
       }
       lastkeyS = keyS;                          // Evita que a
repeticao da leitura do mesmo botao
     }
     delay(1);
 }
 mySerial.println("AT+VTS=\"*,*,*\"");      // Geracao de DTMF na
ligacao ( \" <- Imprime aspas na Serial)
}


// ================================================================ //
//                            TELECOMANDOS                          //
// ================================================================ //


void Lockit() {
 digitalWrite(Lock, HIGH);
 delay(300);
 digitalWrite(Lock, LOW);
 delay(200);
 mySerial.println("AT+VTS=\"9\"");      // Geracao de DTMF na ligacao (
\" <- Imprime aspas na Serial)
 Serial.println("Carro trancado!!");
 startTime = millis();
}


void unLockit() {
 digitalWrite(unLock, HIGH);
 delay(300);
 digitalWrite(unLock, LOW);
 delay(200);
 mySerial.println("AT+VTS=\"9\"");      // Geracao de DTMF na ligacao (
\" <- Imprime aspas na Serial)
 Serial.println("Carro Destrancado!!");
 startTime = millis();
}


void openTrunk() {
 digitalWrite(Trunk, HIGH);
 delay(300);
 digitalWrite(Trunk, LOW);
 delay(200);
 digitalWrite(Trunk, HIGH);
 delay(300);
 digitalWrite(Trunk, LOW);
```

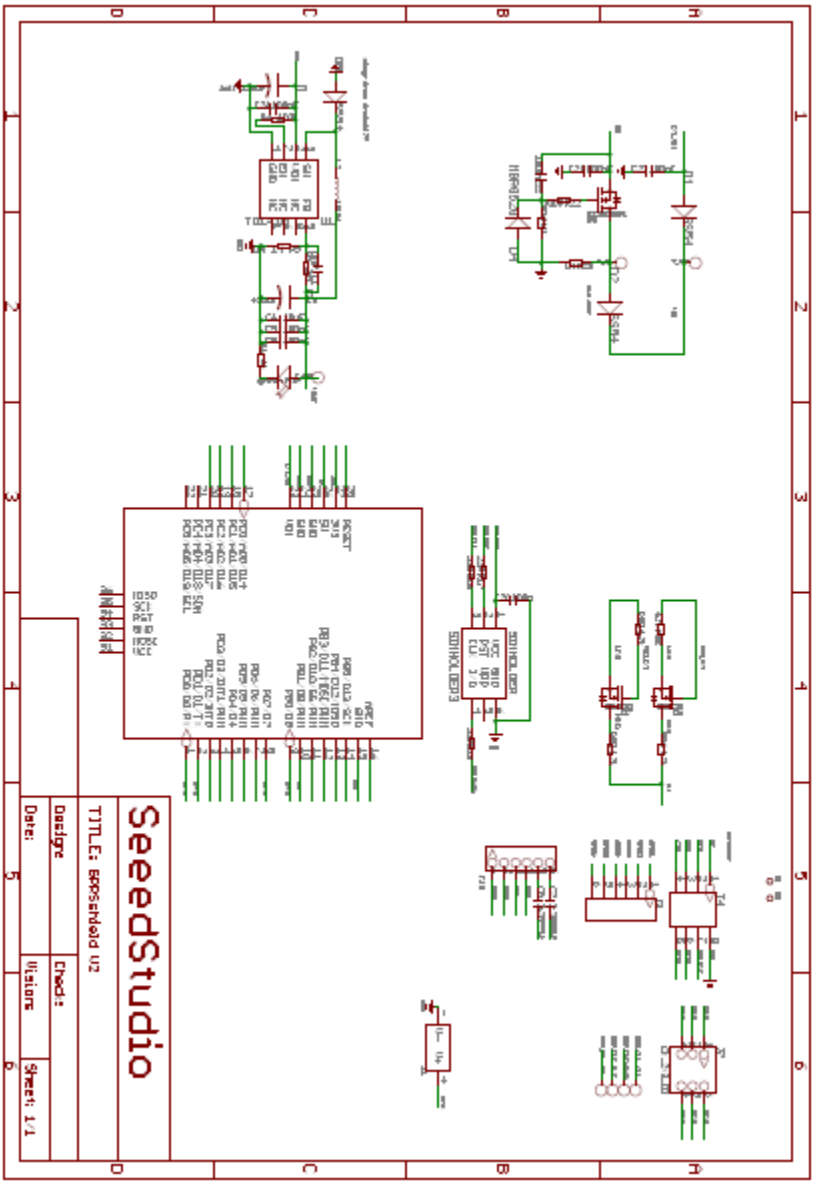
```
 delay(200);
 mySerial.println("AT+VTS=\"9\"");      // Geracao de DTMF na ligacao (
\" <- Imprime aspas na Serial)
 Serial.println("Porta-malas Aberto!");
 startTime = millis();
}


void PANICit() {
 digitalWrite(PANIC, HIGH);
 delay(300);
 digitalWrite(PANIC, LOW);
 delay(100);
 mySerial.println("AT+VTS=\"9\"");      // Geracao de DTMF na ligacao (
\" <- Imprime aspas na Serial)
 Serial.println("PANIC PANIC !!");
 startTime = millis();
}
```

ANEXO II

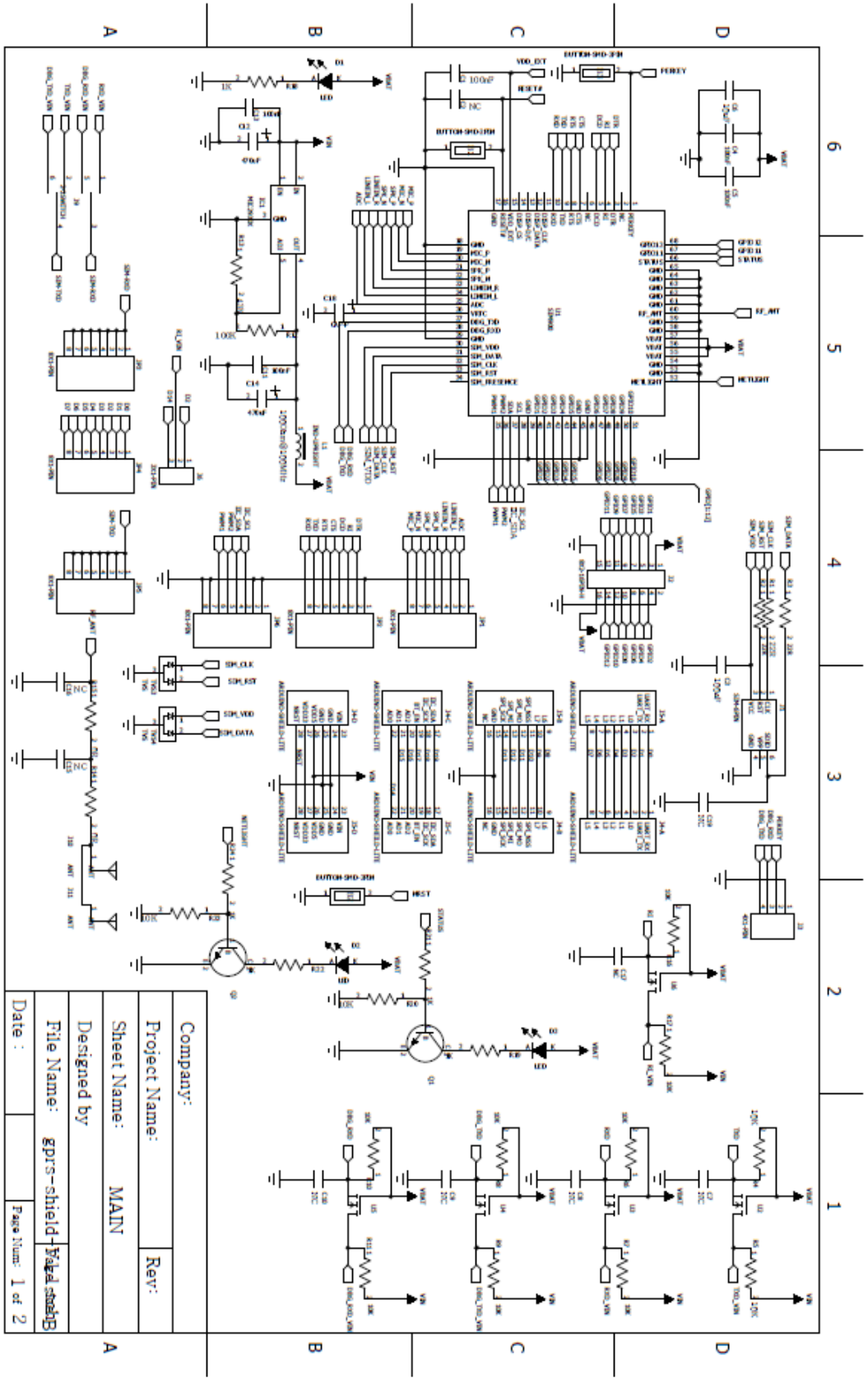


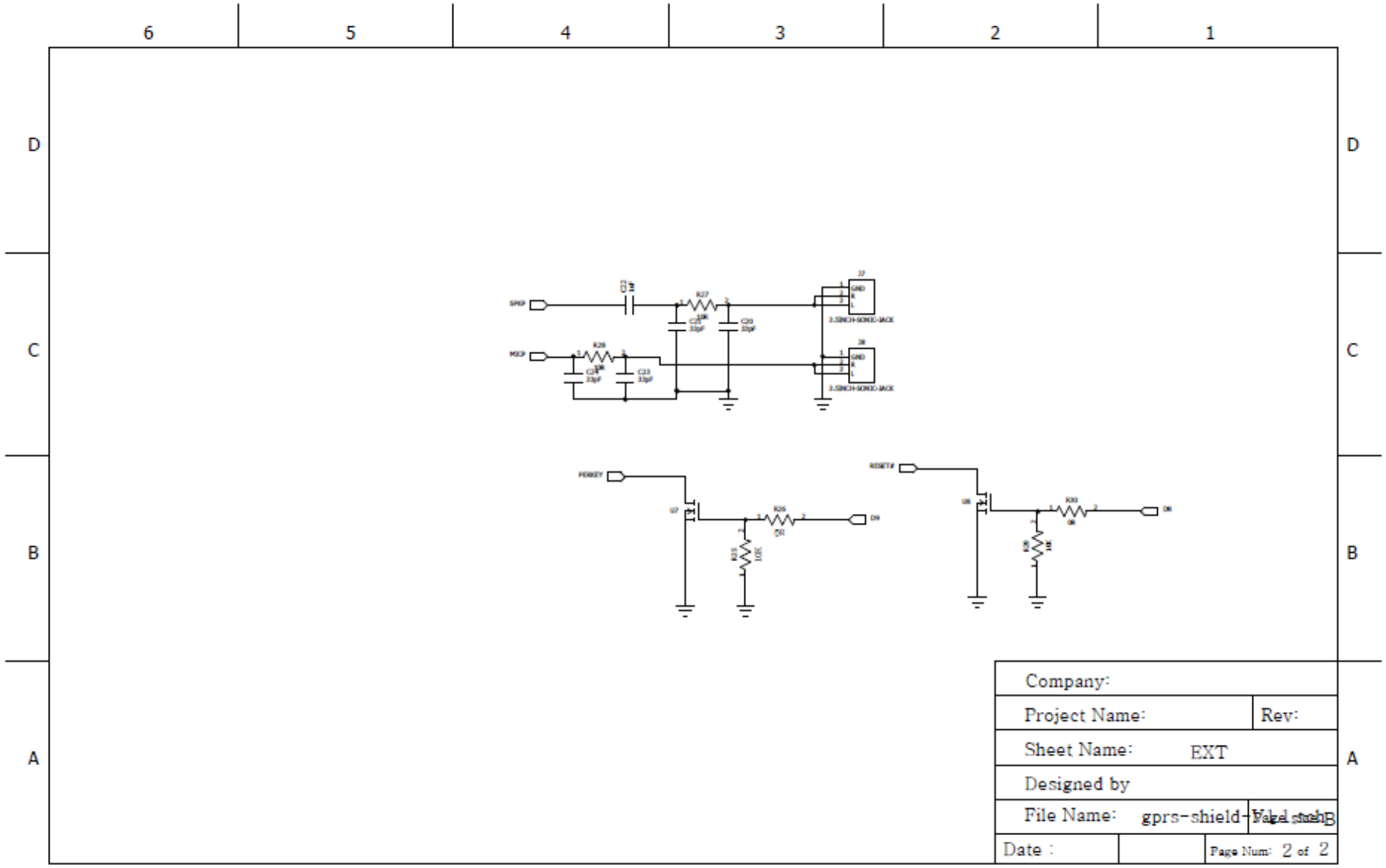
ANEXO III




SeeedStudio

ANEXO IV


Company:
Project Name:
Rev:
Sheet Name: MAIN
Designed by
File Name: gprs-shield-V1.4.sch
Date :
Page Num: 1 of 2


6
5
4
3
2
1
D
C
B
A
SPKP
MICP
3.5INCH-SONIC-JACK
3.5INCH-SONIC-JACK
PERKEY
RESET#
Company:
Project Name:
Rev:
Sheet Name: EXT
Designed by
File Name: gprs-shield-V4.1.sch
Date :
Page Num: 2 of 2